

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

Hossein Movahhedian*

Released 2021-01-27 v0.1c

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

Contents

1	TeXNegar Implementation	3
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	13
1.7	File: texnegar-char-table.lua	17
1.8	File: texnegar.lua	21
1.9	File: texnegar-ini.lua	24
1.10	File: texnegar-luatex-kashida.lua	25
2	Acknowledgments	36
3	Change History	36
	[2020-08-29 v0.1a]	36
	[2020-08-30 v0.1b]	36
	[2021-01-27 v0.1c]	36
	Index	37

1 T_EXNegar 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} {2021-01-27} {0.1c} { 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} {2021-01-27} {0.1c} { 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} {2021-01-27} {0.1c} { 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} {2021-01-27} {0.1c} { 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_glyph_bool
138 \bool_set_false:N \l_texnegar_kashida_leaders_glyph_bool
139 \bool_set_false:N \l_texnegar_kashida_leaders_hrule_bool
140
141 \bool_set_false:N \l_texnegar_ligature_bool
142 \bool_set_false:N \l_texnegar_linebreakpenalty_bool
143 \bool_set_false:N \l_texnegar_hboxrecursion_bool
144 \bool_set_false:N \l_texnegar_vboxrecursion_bool
145 \bool_set_false:N \l_texnegar_color_bool
146
147 \int_set:Nn \l_texnegar_min_penalty_int { 0 }
148 \int_set:Nn \l_texnegar_low_penalty_int { 8 }
149 \int_set:Nn \l_texnegar_med_penalty_int { 15 }
150 \int_set:Nn \l_texnegar_high_penalty_int { 25 }
151 \int_set:Nn \l_texnegar_max_penalty_int { 10000 }
152

```

```

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

```

```

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

```

```

260
261 \msg_new:nnn { texnegar } { luatex-version-is-too-old }
262 {
263   #1:~Your~luatex~is~too~old,~you~need~at~least~version~#2.#3~!
264 }
265
266 \keys_define:nn { texnegar }
267 {
268   Minimal .code:n =
269   {
270     \tl_set:Nn \l_tmpa_tl { #1 }
271     \tl_case:Nn \l_tmpa_tl
272     {
273       \l_texnegar_minimal_off_tl
274       {
275         \bool_set_false:N \l_texnegar_minimal_bool
276       }
277       \l_texnegar_minimal_on_tl
278       {
279         \bool_set_true:N \l_texnegar_minimal_bool
280       }
281     }
282   } ,
283
284   Kashida .code:n =
285   {
286     \tl_set:Nn \l_tmpa_tl { #1 }
287     \tl_case:NnTF \l_tmpa_tl
288     {
289       \l_texnegar_stretch_glyph_tl
290       {
291         \msg_warning:nnn { texnegar } { warning-experimental-feature } { Kashida=gly
292         \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_glyph_tl }
293         \AtBeginDocument
294         {
295           \tl_set:Nx \l_texnegar_main_font_full_tl { \tex_fontname:D \tex_the:D \t
296           \tl_set:Nx \l_texnegar_main_font_name_tl { \l_texnegar_main_font_full_tl
297           \regex_replace_once:nnN { ^"([~/]+)/.* } { \1 } \l_texnegar_main_font_na
298         }
299         \bool_set_true:N \l_texnegar_kashida_fix_bool
300         \bool_set_true:N \l_texnegar_kashida_glyph_bool
301       }
302       \l_texnegar_stretch_leaders_glyph_tl
303       {
304         \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_leaders_glyph_tl
305         \bool_set_true:N \l_texnegar_kashida_fix_bool
306         \bool_set_true:N \l_texnegar_kashida_leaders_glyph_bool
307       }
308       \l_texnegar_stretch_leaders_hruler_tl
309       {
310         \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_leaders_hruler_tl
311         \bool_set_true:N \l_texnegar_kashida_fix_bool
312         \bool_set_true:N \l_texnegar_kashida_leaders_hruler_bool
313       }

```

```

314         \l_texnegar_stretch_off_tl
315         {
316             \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_off_tl }
317             \bool_set_false:N \l_texnegar_kashida_fix_bool
318         }
319         \l_texnegar_stretch_on_tl
320         {
321             \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_leaders_glyph_tl
322             \bool_set_true:N \l_texnegar_kashida_fix_bool
323             \bool_set_true:N \l_texnegar_kashida_leaders_glyph_bool
324         }
325     } { } { \tl_set:Nx \l_texnegar_gap_filler_tl { #1 } }
326     \tl_if_empty:NT \l_texnegar_gap_filler_tl { \msg_error:nn { texnegar } { error-
specify-value-for-kashida-option } }
327 } ,
328
329 linebreakpenalty .code:n =
330 {
331     \int_set:Nn \l_tmpa_int { #1 }
332     \int_case:nnTF \l_tmpa_int
333     {
334         \l_texnegar_min_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
335         \l_texnegar_low_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
336         \l_texnegar_med_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
337         \l_texnegar_high_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
338         \l_texnegar_max_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
339     } { } { \int_set:Nn \l_texnegar_line_break_penalty_int { #1 } }
340     \bool_set_true:N \l_texnegar_linebreakpenalty_bool
341 } ,
342
343 kashidastretch .code:n =
344 {
345     \tl_set:Nn \l_tmpa_tl { #1 }
346     \tl_case:NnTF \l_tmpa_tl
347     {
348         \l_texnegar_fnt_kayhan_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
349         \l_texnegar_fnt_kayhannavaar_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
350         \l_texnegar_fnt_kayhanpook_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
351         \l_texnegar_fnt_kayhansayeh_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
352         \l_texnegar_fnt_khoramshahr_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
353         \l_texnegar_fnt_khorramshahr_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
354         \l_texnegar_fnt_niloofer_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
355         \l_texnegar_fnt_paatch_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
356         \l_texnegar_fnt_riyaz_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
357         \l_texnegar_fnt_roya_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
358         \l_texnegar_fnt_shafigh_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
359         \l_texnegar_fnt_shafighKurd_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
360         \l_texnegar_fnt_shafighUzbek_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
361         \l_texnegar_fnt_shiraz_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
362         \l_texnegar_fnt_sols_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
363         \l_texnegar_fnt_tabriz_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.11
364         \l_texnegar_fnt_titr_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
365         \l_texnegar_fnt_titre_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
366         \l_texnegar_fnt_traffic_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12

```

```

367         \l_texnegar_fnt_vahid_tl      { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
368         \l_texnegar_fnt_vosta_tl      { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
369         \l_texnegar_fnt_yaghut_tl     { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
370         \l_texnegar_fnt_yagut_tl      { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
371         \l_texnegar_fnt_yas_tl        { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
372         \l_texnegar_fnt_yekan_tl      { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
373         \l_texnegar_fnt_yermook_tl    { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
374         \l_texnegar_fnt_zar_tl        { \tl_set:Nn \l_texnegar_skip_default_tl { 0.11
375         \l_texnegar_fnt_ziba_tl       { \tl_set:Nn \l_texnegar_skip_default_tl { 0.11
376         \l_texnegar_fnt_default_tl    { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
377         \l_texnegar_fnt_noskip_tl     { \tl_set:Nn \l_texnegar_skip_default_tl { 0
378     } { } { \tl_set:Nn \l_texnegar_skip_default_tl { #1 } }
379 } ,
380 kashidastretch .default:n = \tl_set:Nn \l_texnegar_skip_default_tl { 0 em plus 0.5 em }
381
382 ligatures .code:n =
383 {
384     \tl_set:Nn \l_tmpa_tl { #1 }
385     \tl_case:NnTF \l_tmpa_tl
386     {
387         \l_texnegar_lig_aalt_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
388         \l_texnegar_lig_ccmp_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
389         \l_texnegar_lig_dlig_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
390         \l_texnegar_lig_fina_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
391         \l_texnegar_lig_init_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
392         \l_texnegar_lig_locl_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
393         \l_texnegar_lig_medi_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
394         \l_texnegar_lig_rlig_tl      { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
395         \l_texnegar_lig_default_tl   { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
396     } { } { \tl_set:Nn \l_texnegar_active_ligs_tl { #1 } }
397     \bool_set_true:N \l_texnegar_ligature_bool
398 } ,
399 ligatures .default:n = \tl_set:Nn \l_texnegar_active_ligs_tl { \l_texnegar_lig_default_t
400
401 color .code:n =
402 {
403     \tl_set:Nn \l_tmpa_tl { #1 }
404     \tl_if_empty:NTF \l_tmpa_tl
405     {
406         \tl_set:Nx \l_texnegar_color_tl { \l_texnegar_col_default_tl }
407     }
408     {
409         \tl_set:Nx \l_texnegar_color_tl { \l_tmpa_tl }
410     }
411     \bool_set_true:N \l_texnegar_color_bool
412     \sys_if_engine luatex:T
413     {
414         \convertcolourspec{named}{\l_texnegar_color_tl}{rgb}\l_texnegar_color_rgb_tl
415         \sys_if_engine luatex:T
416         {
417             \directlua{\l_texnegar_color_rgb_tl = "\l_texnegar_color_rgb_tl"}
418         }
419     }
420 } ,

```

```

421
422 hboxrecursion .code:n =
423 {
424   \tl_set:Nn \l_tmpa_tl { #1 }
425   \tl_case:NnTF \l_tmpa_tl
426   {
427     \l_texnegar_hboxrecursion_off_tl
428     {
429       \bool_set_false:N \l_texnegar_hboxrecursion_bool
430     }
431     \l_texnegar_hboxrecursion_on_tl
432     {
433       \bool_set_true:N \l_texnegar_hboxrecursion_bool
434     }
435   } { } { \bool_set_false:N \l_texnegar_hboxrecursion_bool }
436 } ,
437 hboxrecursion .default:n = \bool_set_true:N \l_texnegar_hboxrecursion_bool ,
438
439 vboxrecursion .code:n =
440 {
441   \tl_set:Nn \l_tmpa_tl { #1 }
442   \tl_case:NnTF \l_tmpa_tl
443   {
444     \l_texnegar_vboxrecursion_off_tl
445     {
446       \bool_set_false:N \l_texnegar_vboxrecursion_bool
447     }
448     \l_texnegar_vboxrecursion_on_tl
449     {
450       \bool_set_true:N \l_texnegar_vboxrecursion_bool
451     }
452   } { } { \bool_set_false:N \l_texnegar_vboxrecursion_bool }
453 } ,
454 vboxrecursion .default:n = \bool_set_true:N \l_texnegar_vboxrecursion_bool ,
455 }
456
457 \ProcessKeysOptions { texnegar }
458
459 \sys_if_engine luatex:T
460 {
461   \NewDocumentCommand \KashidaHMFfixOff {} { \directlua{StopStretching()} }
462   \NewDocumentCommand \KashidaHMFfixOn {} { \directlua{StartStretching()} }
463 }
464
465 \sys_if_engine xetex:T
466 {
467   \NewDocumentCommand \KashidaHMFfixOn {} { \bool_set_true:N \l_texnegar_kashida_fix_bool }
468   \NewDocumentCommand \KashidaHMFfixOff {} { \bool_set_false:N \l_texnegar_kashida_fix_bool }
469 }
470
471 \tex_let:D \KashidaOn \KashidaHMFfixOn
472 \tex_let:D \KashidaOff \KashidaHMFfixOff
473
474 \bool_if:NTF \l_texnegar_kashida_fix_bool

```

```

475 {
476   \tl_if_empty:NT \l_texnegar_skip_default_tl { \tl_set:Nn \l_texnegar_skip_default_tl {
477 }
478 {
479   \tl_set:NV \l_texnegar_skip_default_tl \c_texnegar_skip_a_tl
480 }
481
482 %% % \makeatletter
483 %% % \newif\if@Kashida@on
484 %% Becuase Vafa Khalighi has copied the above code (injecting the character uni+200E) in xepersian.
23.0
485 %% (https://tug.org/svn/texlive/trunk/Master/texmf-dist/tex/xelatex/xepersian/kashida-
xepersian.def?revision=55165&view=co),
486 %% the following line of code is not needed in xepersian anymore.
487 %% % \newif\if@Kashida@XB@fix
488 %% % \makeatother
489
490 \bool_if:NF \l_texnegar_minimal_bool
491 {
492   \input texnegar-luabidi.tex
493 }
494
495 \endinput
496 </texnegar-ini-tex>

```

1.5 File: texnegar-common-kashida.tex

```

497 <*texnegar-common-kashida-tex>
498 \ProvidesExplFile {texnegar-common-kashida.tex} {2021-01-27} {0.1c} { Full implementation of
499
500 \bool_if:NT \l_texnegar_ligature_bool
501 {
502   \clist_new:N \l_texnegar_ligatures_clist
503   \int_new:N \l_texnegar_lig_names_len_int
504   \int_set:Nn \l_texnegar_lig_names_len_int { \clist_count:N \l_texnegar_lig_names_clist }
505   \int_step_inline:nnnn { 1 } { 2 } { \l_texnegar_lig_names_len_int }
506   {
507     \int_set:Nn \l_tmpa_int { #1 }
508     \int_set:Nn \l_tmpb_int { \int_eval:n { \l_tmpa_int + 1 } }
509     \tl_set:Nf \l_tmpa_tl { \clist_item:Nn \l_texnegar_lig_names_clist { \l_tmpa_int } }
510     \clist_set:Nx \l_tmpa_clist { { \clist_item:Nn \l_texnegar_lig_names_clist { \l_tmpb_int } }
511     \bool_if:nT { \tl_if_eq_p:NN \l_texnegar_active_ligs_tl \l_tmpa_tl || \tl_if_eq_p:NN
512       {
513         \clist_put_left:Nx \l_texnegar_ligatures_clist { \l_tmpa_clist }
514       }
515     }
516   \clist_map_inline:Nn \l_texnegar_ligatures_clist
517   {
518     \seq_set_split:Nnn \l_tmpa_seq { = } { #1 }
519     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl { } { }
520     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl { } { }
521     \tl_const:cx { \tl_use:N \l_tmpb_tl } { \char" \l_tmpa_tl \ }
522   }
523 }
524

```



```

577
578 \tex_input:D { texnegar-common-kashida.tex }
579
580 \tl_set:Nn \l_texnegar_use_color_tl
581 {
582   \bool_if:NTF \l_texnegar_color_bool
583   {
584     \colorlet{default}{\l_texnegar_color_tl}
585   }
586   {
587     \colorlet{default}{.}
588   }
589   \color{default}
590 }
591
592 %% Partly adapted from the code provided by David Carlisle in:
593 %% https://tex.stackexchange.com/questions/356709/how-to-know-the-width-and-fill-the-glue-space-between-two-characters-when-using/356721#356721
594 \cs_new:Npn \texnegar_kashida_glyph #1
595 {
596   \bool_if:NT \l_texnegar_kashida_fix_bool
597   {
598     \c_texnegar_lrm_int\tex_penalty:D 10000
599     \mode_leave_vertical:
600     \tex_global:D \tex_advance:D \l_texnegar_counter_int \c_one_int
601
602     \tl_set:Nx \l_texnegar_pos_tl { p\tex_romannumeral:D \l_texnegar_counter_int }
603     \tl_set:Nx \l_texnegar_zref_tl { z\tex_romannumeral:D \l_texnegar_counter_int }
604
605     \zsaveposx{x_i\l_texnegar_zref_tl}
606     \tl_set:Nx \l_tmpa_tl
607     {
608       \iow_now:cx { @auxout }
609       {
610         \token_to_str:N \gdef \exp_after:wN \token_to_str:N \cs:w xi\l_texnegar_pos_tl \cs
611       }
612     }
613     \l_tmpa_tl
614     \skip_horizontal:n { #1 }
615     \zsaveposx{x_f\l_texnegar_zref_tl}
616     \tl_set:Nx \l_tmpa_tl
617     {
618       \iow_now:cx { @auxout }
619       {
620         \token_to_str:N \gdef \exp_after:wN \token_to_str:N \cs:w xf\l_texnegar_pos_tl \cs
621       }
622     }
623     \l_tmpa_tl
624     \exp_after:wN
625     \if_meaning:w
626     \cs:w xi\l_texnegar_pos_tl \cs_end: \tex_relax:D
627   \else:
628     \dim_set:Nn \l_texnegar_diff_pos_dim
629     {

```

```

630         \dim_eval:n { \cs:w xi\l_texnegar_pos_tl \cs_end: sp - \cs:w xf\l_texnegar_pos_tl
631     }
632     \dim_compare:nTF { \l_texnegar_diff_pos_dim == 0sp }
633     { }
634     { \llap { \resizebox { \l_texnegar_diff_pos_dim \tex_relax:D } { \height } { \l_texnegar_pos_tl
635 \fi:
636 }
637 }
638
639 \cs_new:Npn \texnegar_kashida_leaders #1
640 {
641     \bool_if:NT \l_texnegar_kashida_fix_bool
642     {
643         \tl_if_eq:NNTF \l_texnegar_gap_filler_tl \l_texnegar_stretch_leaders_glyph_tl
644         {
645             \tl_set:Nx \l_texnegar_font_full_tl { \tex_fontname:D \tex_the:D \tex_font:D }
646             \tl_set:Nx \l_texnegar_font_name_tl { \l_texnegar_font_full_tl }
647             \tl_set:Nx \l_texnegar_font_init_tl { \l_texnegar_font_name_tl }
648             \regex_replace_once:nnN { ~"[? (HM)[\ \ ](X|F).* } { \1\2 } \l_texnegar_font_init_tl
649             \tl_set:Nn \l_tmpa_tl { HMF }
650             \tl_set:Nn \l_tmpb_tl { HMX }
651             \bool_if:nTF { \str_if_eq_p:NN { \l_texnegar_font_init_tl } { \l_tmpa_tl } || \str_if_eq_p:NN
652             {
653                 \hbox_set:Nn \l_texnegar_ksh_box { \l_texnegar_use_color_tl \XeTeXglyph\XeTeXglyph
654                 \c_texnegar_zwj_int \tex_penalty:D 10000
655                 \tex_leaders:D \copy\l_texnegar_ksh_box \skip_horizontal:n { #1 }
656                 \c_texnegar_zwj_int
657             }
658             {
659                 \msg_error:nnx { texnegar } { hm-series-font-not-found } { \l_texnegar_font_name_tl }
660             }
661         }
662     }
663     %% Partly adapted from the code provided by Jonathan Kew in:
664     %% https://tug.org/pipermail/xetex/2009-February/012307.html.
665     %% Somebody notified me that the code in 'kashida-xepersian.def' from xepersian
666     %% package is an exact copy of Jonathan Kew's code. Being unaware of this, in
667     %% the earlier versions of this package I made a mistake and acknowledged
668     %% Vafa Khalighi instead of Jonathan Kew. A sincere thank you to Jonathan Kew
669     %% for his excellent code.
670     \c_texnegar_lrm_int\c_texnegar_zwj_int
671     {\l_texnegar_use_color_tl\tex_penalty:D 10000
672     \tex_leaders:D \tex_hrulerule:D height \XeTeXglyphheight \c_texnegar_two_int
673     \int_use:N \XeTeXcharglyph \c_texnegar_ksh_int depth \XeTeXglyphdepth \c_texnegar_two_int
674     \int_use:N \XeTeXcharglyph \c_texnegar_ksh_int \skip_horizontal:n { #1 }
675     }
676     \c_texnegar_zwj_int
677 }
678 }
679 }
680
681 \XeTeXinterchartokenstate = 1
682
683 \clist_set:Nn \l_texnegar_a_clist { 0622,0623,0625,0627 } %

```

```

684 \clist_map_inline:Nn \l_texnegar_a_clist
685 {
686   \XeTeXcharclass "#1 \c_texnegar_a_charclass
687 }
688
689 \clist_set:Nn \l_texnegar_d_clist { 0626,0628,062A,062B,062C,062D,062E,0633,0634,0635,0636,0
690 \clist_map_inline:Nn \l_texnegar_d_clist
691 {
692   \XeTeXcharclass "#1 \c_texnegar_d_charclass
693 }
694
695 \clist_set:Nn \l_texnegar_l_clist { 0644 } %
696 \clist_map_inline:Nn \l_texnegar_l_clist
697 {
698   \XeTeXcharclass "#1 \c_texnegar_l_charclass
699 }
700
701 \clist_set:Nn \l_texnegar_r_clist { 0624,0629,062F,0630,0631,0632,0648,0698 } % , , , , , ,
702 \clist_map_inline:Nn \l_texnegar_r_clist
703 {
704   \XeTeXcharclass "#1 \c_texnegar_r_charclass
705 }
706
707 \clist_set:Nn \l_texnegar_y_clist { 0649,064A,06CC } % , ,
708 \clist_map_inline:Nn \l_texnegar_y_clist
709 {
710   \XeTeXcharclass "#1 \c_texnegar_y_charclass
711 }
712
713 \tl_if_eq:NNTF \l_texnegar_gap_filler_tl \l_texnegar_stretch_glyph_tl {
714   \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_y_charclass = {
715     \bool_if:NTF \l_texnegar_kashida_fix_bool
716     { \c_texnegar_zwj_int \texnegar_kashida_glyph \l_texnegar_skip_default_tl \c_texnegar_zwj_int
717     { \c_texnegar_zwj_int \texnegar_kashida_glyph \c_texnegar_skip_a_tl \c_texnegar_zwj_int
718   }
719   \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_y_charclass = {
720     \bool_if:NTF \l_texnegar_kashida_fix_bool
721     { \c_texnegar_zwj_int \texnegar_kashida_glyph \l_texnegar_skip_default_tl \c_texnegar_zwj_int
722     { \c_texnegar_zwj_int \texnegar_kashida_glyph \c_texnegar_skip_a_tl \c_texnegar_zwj_int
723   }
724   \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_d_charclass = { \c_texnegar_zwj_int
725   \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_d_charclass = { \c_texnegar_zwj_int
726   \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_d_charclass = { \c_texnegar_zwj_int
727   \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_l_charclass = { \c_texnegar_zwj_int
728   \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_l_charclass = { \c_texnegar_zwj_int
729   \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_r_charclass = { \c_texnegar_zwj_int
730   \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_a_charclass = { \c_texnegar_zwj_int
731   \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_r_charclass = { \c_texnegar_zwj_int
732   \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_a_charclass = { }
733 }
734 {
735   \bool_if:NTF {
736     \tl_if_eq_p:NN \l_texnegar_gap_filler_tl \l_texnegar_stretch_leaders_glyph_tl ||
737     \tl_if_eq_p:NN \l_texnegar_gap_filler_tl \l_texnegar_stretch_leaders_hruler_tl

```

```

738 }
739 {
740   \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_y_charclass = {
741     \bool_if:NTF \l_texnegar_kashida_fix_bool
742     { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
743     { \texnegar_kashida_leaders \c_texnegar_skip_a_tl }
744   }
745   \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_y_charclass = {
746     \bool_if:NTF \l_texnegar_kashida_fix_bool
747     { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
748     { \texnegar_kashida_leaders \c_texnegar_skip_a_tl }
749   }
750   \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_d_charclass = { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
751   \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_d_charclass = { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
752   \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_d_charclass = { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
753   \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_l_charclass = { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
754   \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_l_charclass = { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
755   \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_r_charclass = { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
756   \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_a_charclass = { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
757   \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_r_charclass = { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
758   \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_a_charclass = { }
759 }
760 {
761   \msg_error:nnx { texnegar } { error-value-not-available-for-kashida-option } { \l_texnegar_kashida_fix_bool }
762 }
763 }
764
765 \endinput
766 </texnegar-xetex-kashida-tex>

```

1.7 File: texnegar-char-table.lua

```

767 < *texnegar-char-table.lua >
768 --
769 -- This is file 'texnegar-char-table.lua',
770 -- generated with the docstrip utility.
771 --
772 -- The original source files were:
773 --
774 -- texnegar.dtx (with options: 'texnegar-char-table-lua')
775 --
776 -- Copyright (C) 2020-2021 Hossein Movahhedian
777 --
778 -- It may be distributed and/or modified under the LaTeX Project Public License,
779 -- version 1.3c or higher (your choice). The latest version of
780 -- this license is at: http://www.latex-project.org/lppl.txt
781 --
782 -- texnegar_char_table = texnegar_char_table or {}
783 -- local texnegar_char_table = texnegar_char_table
784 -- texnegar_char_table.module = {
785 --   name = "texnegar_char_table",
786 --   version = "0.1c",
787 --   date = "2021-01-27",
788 --   description = "Full implementation of kashida feature in XeLaTeX and LuaLaTeX",
789 --   author = "Hossein Movahhedian",

```

```

790     copyright      = "Hossein Movahhedian",
791     license         = "LPPL v1.3c"
792 }
793 --
794 -- -- ^A% texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
795 -- local err, warn, info, log = luatexbase.provides_module(texnegar_char_table.module)
796 -- texnegar_char_table.log      = log or (function (s) luatexbase.module_info("texnegar_char_table", s))
797 -- texnegar_char_table.warning = warn or (function (s) luatexbase.module_warning("texnegar_char_table", s))
798 -- texnegar_char_table.error   = err or (function (s) luatexbase.module_error("texnegar_char_table", s))
799
800 local peCharTableDigit = {
801     [1632] = utf8.char(1632), -- "", utf8.codepoint("") == 1632, "\u{0660}", ARABIC-
INDIC DIGIT ZERO
802     [1633] = utf8.char(1633), -- "", utf8.codepoint("") == 1633, "\u{0661}", ARABIC-
INDIC DIGIT ONE
803     [1634] = utf8.char(1634), -- "", utf8.codepoint("") == 1634, "\u{0662}", ARABIC-
INDIC DIGIT TWO
804     [1635] = utf8.char(1635), -- "", utf8.codepoint("") == 1635, "\u{0663}", ARABIC-
INDIC DIGIT THREE
805     [1636] = utf8.char(1636), -- "", utf8.codepoint("") == 1636, "\u{0664}", ARABIC-
INDIC DIGIT FOUR
806     [1637] = utf8.char(1637), -- "", utf8.codepoint("") == 1637, "\u{0665}", ARABIC-
INDIC DIGIT FIVE
807     [1638] = utf8.char(1638), -- "", utf8.codepoint("") == 1638, "\u{0666}", ARABIC-
INDIC DIGIT SIX
808     [1639] = utf8.char(1639), -- "", utf8.codepoint("") == 1639, "\u{0667}", ARABIC-
INDIC DIGIT SEVEN
809     [1640] = utf8.char(1640), -- "", utf8.codepoint("") == 1640, "\u{0668}", ARABIC-
INDIC DIGIT EIGHT
810     [1641] = utf8.char(1641), -- "", utf8.codepoint("") == 1641, "\u{0669}", ARABIC-
INDIC DIGIT NINE
811     [1780] = utf8.char(1780), -- "", utf8.codepoint("") == 1780, "\u{06F4}", EXTENDED ARABIC-
INDIC DIGIT FOUR
812     [1781] = utf8.char(1781), -- "", utf8.codepoint("") == 1781, "\u{06F5}", EXTENDED ARABIC-
INDIC DIGIT FIVE
813     [1782] = utf8.char(1782), -- "", utf8.codepoint("") == 1782, "\u{06F6}", EXTENDED ARABIC-
INDIC DIGIT SIX
814 }
815
816 local peCharTablePunctuation = {
817     [1548] = utf8.char(1548), -- "", utf8.codepoint("") == 1548, "\u{060C}", ARABIC COMMA
818     [1549] = utf8.char(1549), -- "", utf8.codepoint("") == 1549, "\u{060D}", ARABIC DATE SEPARATOR
819     [1563] = utf8.char(1563), -- "", utf8.codepoint("") == 1563, "\u{061B}", ARABIC SEMICOLON
820     [1567] = utf8.char(1567), -- "", utf8.codepoint("") == 1567, "\u{061F}", ARABIC QUESTION MARK
821     [1642] = utf8.char(1642), -- "", utf8.codepoint("") == 1642, "\u{066A}", ARABIC PERCENT SIGN
822     [1643] = utf8.char(1643), -- "", utf8.codepoint("") == 1643, "\u{066B}", ARABIC DECIMAL SIGN
823     [1644] = utf8.char(1644), -- "", utf8.codepoint("") == 1644, "\u{066C}", ARABIC THOUSANDS SEPARATOR
824     [1645] = utf8.char(1645), -- "", utf8.codepoint("") == 1645, "\u{066D}", ARABIC FIVE TENTHS SIGN
825 }
826
827 local peCharTable = {
828     [1569] = utf8.char(1569), -- "", utf8.codepoint("") == 1569, "\u{0621}", ARABIC LETTER ALEF
829     [1570] = utf8.char(1570), -- "", utf8.codepoint("") == 1570, "\u{0622}", ARABIC LETTER BA
830     [1571] = utf8.char(1571), -- "", utf8.codepoint("") == 1571, "\u{0623}", ARABIC LETTER TA

```

```

831 [1572] = utf8.char(1572), -- "", utf8.codepoint("") == 1572, "\u{0624}", ARABIC LETTE
832 [1573] = utf8.char(1573), -- "", utf8.codepoint("") == 1573, "\u{0625}", ARABIC LETTE
833 [1574] = utf8.char(1574), -- "", utf8.codepoint("") == 1574, "\u{0626}", ARABIC LETTE
834 [1575] = utf8.char(1575), -- "", utf8.codepoint("") == 1575, "\u{0627}", ARABIC LETTE
835 [1576] = utf8.char(1576), -- "", utf8.codepoint("") == 1576, "\u{0628}", ARABIC LETTE
836 [1577] = utf8.char(1577), -- "", utf8.codepoint("") == 1577, "\u{0629}", ARABIC LETTE
837 [1578] = utf8.char(1578), -- "", utf8.codepoint("") == 1578, "\u{062A}", ARABIC LETTE
838 [1579] = utf8.char(1579), -- "", utf8.codepoint("") == 1579, "\u{062B}", ARABIC LETTE
839 [1580] = utf8.char(1580), -- "", utf8.codepoint("") == 1580, "\u{062C}", ARABIC LETTE
840 [1581] = utf8.char(1581), -- "", utf8.codepoint("") == 1581, "\u{062D}", ARABIC LETTE
841 [1582] = utf8.char(1582), -- "", utf8.codepoint("") == 1582, "\u{062E}", ARABIC LETTE
842 [1583] = utf8.char(1583), -- "", utf8.codepoint("") == 1583, "\u{062F}", ARABIC LETTE
843 [1584] = utf8.char(1584), -- "", utf8.codepoint("") == 1584, "\u{0630}", ARABIC LETTE
844 [1585] = utf8.char(1585), -- "", utf8.codepoint("") == 1585, "\u{0631}", ARABIC LETTE
845 [1586] = utf8.char(1586), -- "", utf8.codepoint("") == 1586, "\u{0632}", ARABIC LETTE
846 [1587] = utf8.char(1587), -- "", utf8.codepoint("") == 1587, "\u{0633}", ARABIC LETTE
847 [1588] = utf8.char(1588), -- "", utf8.codepoint("") == 1588, "\u{0634}", ARABIC LETTE
848 [1589] = utf8.char(1589), -- "", utf8.codepoint("") == 1589, "\u{0635}", ARABIC LETTE
849 [1590] = utf8.char(1590), -- "", utf8.codepoint("") == 1590, "\u{0636}", ARABIC LETTE
850 [1591] = utf8.char(1591), -- "", utf8.codepoint("") == 1591, "\u{0637}", ARABIC LETTE
851 [1592] = utf8.char(1592), -- "", utf8.codepoint("") == 1592, "\u{0638}", ARABIC LETTE
852 [1593] = utf8.char(1593), -- "", utf8.codepoint("") == 1593, "\u{0639}", ARABIC LETTE
853 [1594] = utf8.char(1594), -- "", utf8.codepoint("") == 1594, "\u{063A}", ARABIC LETTE
854 [1601] = utf8.char(1601), -- "", utf8.codepoint("") == 1601, "\u{0641}", ARABIC LETTE
855 [1602] = utf8.char(1602), -- "", utf8.codepoint("") == 1602, "\u{0642}", ARABIC LETTE
856 [1603] = utf8.char(1603), -- "", utf8.codepoint("") == 1603, "\u{0643}", ARABIC LETTE
857 [1604] = utf8.char(1604), -- "", utf8.codepoint("") == 1604, "\u{0644}", ARABIC LETTE
858 [1605] = utf8.char(1605), -- "", utf8.codepoint("") == 1605, "\u{0645}", ARABIC LETTE
859 [1606] = utf8.char(1606), -- "", utf8.codepoint("") == 1606, "\u{0646}", ARABIC LETTE
860 [1607] = utf8.char(1607), -- "", utf8.codepoint("") == 1607, "\u{0647}", ARABIC LETTE
861 [1608] = utf8.char(1608), -- "", utf8.codepoint("") == 1608, "\u{0648}", ARABIC LETTE
862 [1609] = utf8.char(1609), -- "", utf8.codepoint("") == 1609, "\u{0649}", ARABIC LETTE
863 [1610] = utf8.char(1610), -- "", utf8.codepoint("") == 1610, "\u{064A}", ARABIC LETTE
864 [1662] = utf8.char(1662), -- "", utf8.codepoint("") == 1662, "\u{067E}", ARABIC LETTE
865 [1670] = utf8.char(1670), -- "", utf8.codepoint("") == 1670, "\u{0686}", ARABIC LETTE
866 [1688] = utf8.char(1688), -- "", utf8.codepoint("") == 1688, "\u{0698}", ARABIC LETTE
867 [1705] = utf8.char(1705), -- "", utf8.codepoint("") == 1705, "\u{06A9}", ARABIC LETTE
868 [1706] = utf8.char(1706), -- "", utf8.codepoint("") == 1706, "\u{06AA}", ARABIC LETTE
869 [1711] = utf8.char(1711), -- "", utf8.codepoint("") == 1711, "\u{06AF}", ARABIC LETTE
870 [1726] = utf8.char(1726), -- "", utf8.codepoint("") == 1726, "\u{06BE}", ARABIC LETTE
871 [1728] = utf8.char(1728), -- "", utf8.codepoint("") == 1728, "\u{06C0}", ARABIC LETTE
872 [1740] = utf8.char(1740), -- "", utf8.codepoint("") == 1740, "\u{06CC}", ARABIC LETTE
873 [1749] = utf8.char(1749), -- "", utf8.codepoint("") == 1740, "\u{06D5}", ARABIC LETTE
874 [65275] = utf8.char(65275), -- "", utf8.codepoint("") == 65275, "\u{FEFB}", ARABIC LIGA
875 [65276] = utf8.char(65276), -- "", utf8.codepoint("") == 65276, "\u{FEFC}", ARABIC LIGA
876 }
877
878 local peCharTableInitial = {
879 [64344] = utf8.char(64344), -- "", utf8.codepoint("") == 64344, "\u{FB58}", INITIAL FOR
880 [64380] = utf8.char(64380), -- "", utf8.codepoint("") == 64380, "\u{FB7C}", INITIAL FOR
881 [64400] = utf8.char(64400), -- "", utf8.codepoint("") == 64400, "\u{FB90}", INITIAL FOR
882 [64404] = utf8.char(64404), -- "", utf8.codepoint("") == 64404, "\u{FB94}", INITIAL FOR
883 [64510] = utf8.char(64510), -- "", utf8.codepoint("") == 64510, "\u{FBFE}", INITIAL FOR
884 [65169] = utf8.char(65169), -- "", utf8.codepoint("") == 65169, "\u{FE91}", INITIAL FOR

```

```

885 [65175] = utf8.char(65175), -- "", utf8.codepoint("") == 65175, "\u{FE97}", INITIAL FOR
886 [65179] = utf8.char(65179), -- "", utf8.codepoint("") == 65179, "\u{FE9B}", INITIAL FOR
887 [65183] = utf8.char(65183), -- "", utf8.codepoint("") == 65183, "\u{FE9F}", INITIAL FOR
888 [65187] = utf8.char(65187), -- "", utf8.codepoint("") == 65187, "\u{FEA3}", INITIAL FOR
889 [65191] = utf8.char(65191), -- "", utf8.codepoint("") == 65191, "\u{FEA7}", INITIAL FOR
890 [65203] = utf8.char(65203), -- "", utf8.codepoint("") == 65203, "\u{FEB3}", INITIAL FOR
891 [65207] = utf8.char(65207), -- "", utf8.codepoint("") == 65207, "\u{FEB7}", INITIAL FOR
892 [65211] = utf8.char(65211), -- "", utf8.codepoint("") == 65211, "\u{FEBB}", INITIAL FOR
893 [65215] = utf8.char(65215), -- "", utf8.codepoint("") == 65215, "\u{FEBF}", INITIAL FOR
894 [65219] = utf8.char(65219), -- "", utf8.codepoint("") == 65219, "\u{FEC3}", INITIAL FOR
895 [65223] = utf8.char(65223), -- "", utf8.codepoint("") == 65223, "\u{FEC7}", INITIAL FOR
896 [65227] = utf8.char(65227), -- "", utf8.codepoint("") == 65227, "\u{FECB}", INITIAL FOR
897 [65231] = utf8.char(65231), -- "", utf8.codepoint("") == 65231, "\u{FECF}", INITIAL FOR
898 [65235] = utf8.char(65235), -- "", utf8.codepoint("") == 65235, "\u{FED3}", INITIAL FOR
899 [65239] = utf8.char(65239), -- "", utf8.codepoint("") == 65239, "\u{FED7}", INITIAL FOR
900 [65243] = utf8.char(65243), -- "", utf8.codepoint("") == 65243, "\u{FEDB}", INITIAL FOR
901 [65247] = utf8.char(65247), -- "", utf8.codepoint("") == 65247, "\u{FEDF}", INITIAL FOR
902 [65251] = utf8.char(65251), -- "", utf8.codepoint("") == 65251, "\u{FEE3}", INITIAL FOR
903 [65255] = utf8.char(65255), -- "", utf8.codepoint("") == 65255, "\u{FEE7}", INITIAL FOR
904 [65259] = utf8.char(65259), -- "", utf8.codepoint("") == 65259, "\u{FEEB}", INITIAL FOR
905 [65267] = utf8.char(65267), -- "", utf8.codepoint("") == 65267, "\u{FEF3}", INITIAL FOR
906 }
907
908 local peCharTableMedial = {
909 [1600] = utf8.char(1600), -- "", utf8.codepoint("") == 1600, "\u{0640}", ARABIC TATW
910 [64345] = utf8.char(64345), -- "", utf8.codepoint("") == 64345, "\u{FB59}", MEDIAL FORM
911 [64381] = utf8.char(64381), -- "", utf8.codepoint("") == 64381, "\u{FB7D}", MEDIAL FORM
912 [64401] = utf8.char(64401), -- "", utf8.codepoint("") == 64401, "\u{FB91}", MEDIAL FORM
913 [64405] = utf8.char(64405), -- "", utf8.codepoint("") == 64405, "\u{FB95}", MEDIAL FORM
914 [64425] = utf8.char(64425), -- "", utf8.codepoint("") == 64425, "\u{FBA9}", MEDIAL FORM
915 [64429] = utf8.char(64429), -- "", utf8.codepoint("") == 64429, "\u{FBAD}", MEDIAL FORM
916 [64511] = utf8.char(64511), -- "", utf8.codepoint("") == 64511, "\u{FBFF}", MEDIAL FORM
917 [65170] = utf8.char(65170), -- "", utf8.codepoint("") == 65170, "\u{FE92}", MEDIAL FORM
918 [65176] = utf8.char(65176), -- "", utf8.codepoint("") == 65176, "\u{FE98}", MEDIAL FORM
919 [65180] = utf8.char(65180), -- "", utf8.codepoint("") == 65180, "\u{FE9C}", MEDIAL FORM
920 [65184] = utf8.char(65184), -- "", utf8.codepoint("") == 65184, "\u{FEA0}", MEDIAL FORM
921 [65188] = utf8.char(65188), -- "", utf8.codepoint("") == 65188, "\u{FEA4}", MEDIAL FORM
922 [65192] = utf8.char(65192), -- "", utf8.codepoint("") == 65192, "\u{FEA8}", MEDIAL FORM
923 [65204] = utf8.char(65204), -- "", utf8.codepoint("") == 65204, "\u{FEB4}", MEDIAL FORM
924 [65208] = utf8.char(65208), -- "", utf8.codepoint("") == 65208, "\u{FEB8}", MEDIAL FORM
925 [65212] = utf8.char(65212), -- "", utf8.codepoint("") == 65212, "\u{FEBC}", MEDIAL FORM
926 [65216] = utf8.char(65216), -- "", utf8.codepoint("") == 65216, "\u{FEC0}", MEDIAL FORM
927 [65220] = utf8.char(65220), -- "", utf8.codepoint("") == 65220, "\u{FEC4}", MEDIAL FORM
928 [65224] = utf8.char(65224), -- "", utf8.codepoint("") == 65224, "\u{FEC8}", MEDIAL FORM
929 [65228] = utf8.char(65228), -- "", utf8.codepoint("") == 65228, "\u{FECC}", MEDIAL FORM
930 [65232] = utf8.char(65232), -- "", utf8.codepoint("") == 65232, "\u{FED0}", MEDIAL FORM
931 [65236] = utf8.char(65236), -- "", utf8.codepoint("") == 65236, "\u{FED4}", MEDIAL FORM
932 [65240] = utf8.char(65240), -- "", utf8.codepoint("") == 65240, "\u{FED8}", MEDIAL FORM
933 [65244] = utf8.char(65244), -- "", utf8.codepoint("") == 65244, "\u{FEDC}", MEDIAL FORM
934 [65248] = utf8.char(65248), -- "", utf8.codepoint("") == 65248, "\u{FEE0}", MEDIAL FORM
935 [65252] = utf8.char(65252), -- "", utf8.codepoint("") == 65252, "\u{FEE4}", MEDIAL FORM
936 [65256] = utf8.char(65256), -- "", utf8.codepoint("") == 65256, "\u{FEE8}", MEDIAL FORM
937 [65260] = utf8.char(65260), -- "", utf8.codepoint("") == 65260, "\u{FEEC}", MEDIAL FORM
938 [65268] = utf8.char(65268), -- "", utf8.codepoint("") == 65268, "\u{FEF4}", MEDIAL FORM

```

```

939 }
940
941 local peCharTableFinal = {
942   [64343] = utf8.char(64343), -- "", utf8.codepoint("") == 64343, "\u{FB57}", FINAL FORM
943   [64379] = utf8.char(64379), -- "", utf8.codepoint("") == 64379, "\u{FB7B}", FINAL FORM
944   [64395] = utf8.char(64395), -- "", utf8.codepoint("") == 64395, "\u{FB8B}", FINAL FORM
945   [64399] = utf8.char(64399), -- "", utf8.codepoint("") == 64399, "\u{FB8F}", FINAL FORM
946   [64403] = utf8.char(64403), -- "", utf8.codepoint("") == 64403, "\u{FB93}", FINAL FORM
947   [64421] = utf8.char(64421), -- "", utf8.codepoint("") == 64421, "\u{FBA5}", FINAL FORM
948   [64509] = utf8.char(64509), -- "", utf8.codepoint("") == 64509, "\u{FBFD}", FINAL FORM
949   [65166] = utf8.char(65166), -- "", utf8.codepoint("") == 65166, "\u{FE8E}", FINAL FORM
950   [65168] = utf8.char(65168), -- "", utf8.codepoint("") == 65168, "\u{FE90}", FINAL FORM
951   [65172] = utf8.char(65172), -- "", utf8.codepoint("") == 65172, "\u{FE94}", FINAL FORM
952   [65174] = utf8.char(65174), -- "", utf8.codepoint("") == 65174, "\u{FE96}", FINAL FORM
953   [65178] = utf8.char(65178), -- "", utf8.codepoint("") == 65178, "\u{FE9A}", FINAL FORM
954   [65182] = utf8.char(65182), -- "", utf8.codepoint("") == 65182, "\u{FE9E}", FINAL FORM
955   [65186] = utf8.char(65186), -- "", utf8.codepoint("") == 65186, "\u{FEA2}", FINAL FORM
956   [65190] = utf8.char(65190), -- "", utf8.codepoint("") == 65190, "\u{FEA6}", FINAL FORM
957   [65194] = utf8.char(65194), -- "", utf8.codepoint("") == 65194, "\u{FEAA}", FINAL FORM
958   [65196] = utf8.char(65196), -- "", utf8.codepoint("") == 65196, "\u{FEAC}", FINAL FORM
959   [65198] = utf8.char(65198), -- "", utf8.codepoint("") == 65198, "\u{FEAE}", FINAL FORM
960   [65200] = utf8.char(65200), -- "", utf8.codepoint("") == 65200, "\u{FEB0}", FINAL FORM
961   [65202] = utf8.char(65202), -- "", utf8.codepoint("") == 65202, "\u{FEB2}", FINAL FORM
962   [65206] = utf8.char(65206), -- "", utf8.codepoint("") == 65206, "\u{FEB6}", FINAL FORM
963   [65210] = utf8.char(65210), -- "", utf8.codepoint("") == 65210, "\u{FEB8}", FINAL FORM
964   [65214] = utf8.char(65214), -- "", utf8.codepoint("") == 65214, "\u{FEBE}", FINAL FORM
965   [65218] = utf8.char(65218), -- "", utf8.codepoint("") == 65218, "\u{FEC2}", FINAL FORM
966   [65222] = utf8.char(65222), -- "", utf8.codepoint("") == 65222, "\u{FEC6}", FINAL FORM
967   [65226] = utf8.char(65226), -- "", utf8.codepoint("") == 65226, "\u{FECA}", FINAL FORM
968   [65230] = utf8.char(65230), -- "", utf8.codepoint("") == 65230, "\u{FECE}", FINAL FORM
969   [65234] = utf8.char(65234), -- "", utf8.codepoint("") == 65234, "\u{FED2}", FINAL FORM
970   [65238] = utf8.char(65238), -- "", utf8.codepoint("") == 65238, "\u{FED6}", FINAL FORM
971   [65242] = utf8.char(65242), -- "", utf8.codepoint("") == 65242, "\u{FEDA}", FINAL FORM
972   [65246] = utf8.char(65246), -- "", utf8.codepoint("") == 65246, "\u{FEDE}", FINAL FORM
973   [65250] = utf8.char(65250), -- "", utf8.codepoint("") == 65250, "\u{FEE2}", FINAL FORM
974   [65254] = utf8.char(65254), -- "", utf8.codepoint("") == 65254, "\u{FEE6}", FINAL FORM
975   [65258] = utf8.char(65258), -- "", utf8.codepoint("") == 65258, "\u{FEEA}", FINAL FORM
976   [65262] = utf8.char(65262), -- "", utf8.codepoint("") == 65262, "\u{FEEE}", FINAL FORM
977   [65264] = utf8.char(65264), -- "", utf8.codepoint("") == 65264, "\u{FEF0}", FINAL FORM
978   [65266] = utf8.char(65266), -- "", utf8.codepoint("") == 65266, "\u{FEF2}", FINAL FORM
979   [65276] = utf8.char(65276), -- "", utf8.codepoint("") == 65276, "\u{FEFC}", FINAL FORM
980 }
981
982 return peCharTableInitial, peCharTableMedial, peCharTableFinal
983 --
984 --
985 -- End of file 'texnegar-char-table.lua'.
986 </texnegar-char-table.lua>

```

1.8 File: texnegar.lua

```

987 <!--texnegar.lua>
988 --
989 -- This is file 'texnegar.lua',
990 -- generated with the docstrip utility.

```

```

991 --
992 -- The original source files were:
993 --
994 -- texnegar.dtx (with options: 'texnegar-lua')
995 --
996 -- Copyright (C) 2020-2021 Hossein Movahhedian
997 --
998 -- It may be distributed and/or modified under the LaTeX Project Public License,
999 -- version 1.3c or higher (your choice). The latest version of
1000 -- this license is at: http://www.latex-project.org/lppl.txt
1001 --
1002 -- texnegar = texnegar or {}
1003 -- local texnegar = texnegar
1004 -- texnegar.module = {
1005 --     name = "texnegar",
1006 --     version = "0.1c",
1007 --     date = "2021-01-27",
1008 --     description = "Full implementation of kashida feature in XeLaTeX and LuaLaTeX",
1009 --     author = "Hossein Movahhedian",
1010 --     copyright = "Hossein Movahhedian",
1011 --     license = "LPPL v1.3c"
1012 -- }
1013 --
1014 -- ^A%% texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
1015 -- local err, warn, info, log = luatexbase.provides_module(texnegar.module)
1016 -- texnegar.log = log or (function (s) luatexbase.module_info("texnegar", s) end)
1017 -- texnegar.warning = warn or (function (s) luatexbase.module_warning("texnegar", s) end)
1018 -- texnegar.error = err or (function (s) luatexbase.module_error("texnegar", s) end)
1019
1020 local debug_getinfo = debug.getinfo
1021 local string_format = string.format
1022
1023 function TableLength(t)
1024     local i = 0
1025     for _ in pairs(t) do
1026         i = i + 1
1027     end
1028     return i
1029 end
1030
1031 tex.enableprimitives('', tex.extraprimitives ())
1032
1033 local range_tble = {
1034     [1536] = 1791,
1035     [1872] = 1919,
1036     [2208] = 2274,
1037     [8204] = 8297,
1038     [64336] = 65023,
1039     [65136] = 65279,
1040     [126464] = 126719,
1041     [983040] = 1048575
1042 }
1043
1044 local tbl_fonts_used = { }

```

```

1045 local tbl_fonts_chars = { }
1046 local tbl_fonts_chars_init = { }
1047 local tbl_fonts_chars_medi = { }
1048 local tbl_fonts_chars_fina = { }
1049
1050 local pattern_list = {
1051     "%.(ini)t?$", "%.(ini)t?%..*",
1052     "%.(med)i?$", "%.(med)i?%..*",
1053     "%.(fin)a?$", "%.(fin)a?%..*",
1054
1055     "%_(ini)t?$", "%_(ini)t?_.*",
1056     "%_(med)i?$", "%_(med)i?_.*",
1057     "%_(fin)a?$", "%_(fin)a?_.*",
1058
1059     "%_(AltIni)t?[0-9]?$", "%_(AltIni)t?[0-9]?_.*",
1060     "%_(AltMed)i?[0-9]?$", "%_(AltMed)i?[0-9]?_.*",
1061     "%_(AltFin)a?[0-9]?$", "%_(AltFin)a?[0-9]?_.*",
1062 }
1063
1064 function GetFontsChars()
1065     local funcName = debug_getinfo(1).name
1066     local funcNparams = debug_getinfo(1).nparams
1067
1068     for f_num = 1, font.max() do
1069         local f_tmp = font.fonts[f_num]
1070         if f_tmp then
1071             local f_tmp_fontname = f_tmp.fontname
1072             if f_tmp_fontname then
1073                 local f_id_tmp = font.getfont(f_num)
1074                 local f_fontname_tmp = f_id_tmp.fontname
1075                 local f_filename_tmp = f_id_tmp.filename
1076                 if not tbl_fonts_used[f_fontname_tmp] then
1077                     tbl_fonts_used[f_fontname_tmp] = {f_filename_tmp, f_id_tmp}
1078                 end
1079             end
1080         end
1081     end
1082
1083     for f_fontname, v in pairs(tbl_fonts_used) do
1084         f_filename = v[1]
1085         f_id = v[2]
1086         if not tbl_fonts_chars[f_fontname] then
1087             tbl_fonts_chars[f_fontname] = { }
1088             tbl_fonts_chars_init[f_fontname] = { }
1089             tbl_fonts_chars_medi[f_fontname] = { }
1090             tbl_fonts_chars_fina[f_fontname] = { }
1091             local f = fontloader.open(f_filename)
1092             local char_name
1093             local char_unicode
1094             local char_class
1095             for k, v in pairs(range_tble) do
1096                 for glyph_idx = k, v do
1097                     if f_id.characters[glyph_idx] then
1098                         char_name = f.glyphs[f_id.characters[glyph_idx].index].name

```

```

1099         char_unicode = f.glyphs[f_id.characters[glyph_idx].index].unicode
1100         char_class    = f.glyphs[f_id.characters[glyph_idx].index].class
1101         if not tbl_fonts_chars[f_fontname][glyph_idx] then
1102             if string.match(f_fontname, "^(Amiri).*)" == "Amiri" and char_n
1103                 current_kashida_unicode = glyph_idx
1104             end
1105             tbl_fonts_chars[f_fontname][glyph_idx] = {char_name, char_unicono
1106             for _, pattern in ipairs( pattern_list ) do
1107                 local pos_alt = string.match(char_name, pattern)
1108                 if pos_alt == 'ini' or pos_alt == 'AltIni' then
1109                     tbl_fonts_chars_init[f_fontname][glyph_idx] = {char_name
1110                 elseif pos_alt == 'med' or pos_alt == 'AltMed' then
1111                     tbl_fonts_chars_medi[f_fontname][glyph_idx] = {char_name
1112                 elseif pos_alt == 'fin' or pos_alt == 'AltFin' then
1113                     tbl_fonts_chars_fina[f_fontname][glyph_idx] = {char_name
1114                 end
1115             end
1116         end
1117     end
1118 end
1119     end
1120     fontloader.close(f)
1121 end
1122 end
1123 return tbl_fonts_used, tbl_fonts_chars, tbl_fonts_chars_init, tbl_fonts_chars_medi, tbl
1124 end
1125
1126 dofile(kpse.find_file("texnegar-ini.lua"))
1127 --
1128 --
1129 -- End of file 'texnegar.lua'.
1130 </texnegar-lua>

```

1.9 File: texnegar-ini.lua

```

1131 <{*texnegar-ini-lua>
1132 --
1133 -- This is file 'texnegar-ini.lua',
1134 -- generated with the docstrip utility.
1135 --
1136 -- The original source files were:
1137 --
1138 -- texnegar.dtx (with options: 'texnegar-ini-lua')
1139 --
1140 -- Copyright (C) 2020-2021 Hossein Movahhedian
1141 --
1142 -- It may be distributed and/or modified under the LaTeX Project Public License,
1143 -- version 1.3c or higher (your choice). The latest version of
1144 -- this license is at: http://www.latex-project.org/lppl.txt
1145 --
1146 -- texnegar_ini      = texnegar_ini or {}
1147 -- local texnegar_ini = texnegar_ini
1148 -- texnegar_ini.module = {
1149 --     name      = "texnegar_ini",
1150 --     version   = "0.1c",

```

```

1151 --      date          = "2021-01-27",
1152 --      description    = "Full implementation of kashida feature in XeLaTeX and LuaLaTeX",
1153 --      author         = "Hossein Movahhedian",
1154 --      copyright      = "Hossein Movahhedian",
1155 --      license        = "LPPL v1.3c"
1156 -- }
1157 --
1158 -- -- ^A%% texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
1159 -- local err, warn, info, log = luatexbase.provides_module(texnegar_ini.module)
1160 -- texnegar_ini.log      = log or (function (s) luatexbase.module_info("texnegar_ini", s)
1161 -- texnegar_ini.warning = warn or (function (s) luatexbase.module_warning("texnegar_ini", s)
1162 -- texnegar_ini.error   = err or (function (s) luatexbase.module_error("texnegar_ini", s)
1163
1164 c_true_bool = token.create("c_true_bool")
1165
1166 l_texnegar_color_bool = token.create("l_texnegar_color_bool")
1167
1168 if l_texnegar_color_bool.mode == c_true_bool.mode then
1169     color_tbl = color_tbl or {}
1170     for item in l_texnegar_color_rgb_tl:gmatch("[^,%s]+") do
1171         table.insert(color_tbl, item)
1172     end
1173 end
1174
1175 dofile(kpse.find_file("texnegar-luatex-kashida.lua"))
1176 --
1177 --
1178 -- End of file 'texnegar-ini.lua'.
1179 </texnegar-ini-lua>

```

1.10 File: texnegar-luatex-kashida.lua

```

1180 <*texnegar-luatex-kashida-lua>
1181 --
1182 -- This is file 'texnegar-luatex-kashida.lua',
1183 -- generated with the docstrip utility.
1184 --
1185 -- The original source files were:
1186 --
1187 -- texnegar.dtx (with options: 'texnegar-luatex-kashida-lua')
1188 --
1189 -- Copyright (C) 2020-2021 Hossein Movahhedian
1190 --
1191 -- It may be distributed and/or modified under the LaTeX Project Public License,
1192 -- version 1.3c or higher (your choice). The latest version of
1193 -- this license is at: http://www.latex-project.org/lppl.txt
1194 --
1195 -- texnegar_luatex_kashida = texnegar_luatex_kashida or {}
1196 -- local texnegar_luatex_kashida = texnegar_luatex_kashida
1197 -- texnegar_luatex_kashida.module = {
1198 --     name          = "texnegar_luatex_kashida",
1199 --     version       = "0.1c",
1200 --     date          = "2021-01-27",
1201 --     description    = "Full implementation of kashida feature in XeLaTeX and L
1202 --     author        = "Hossein Movahhedian",

```

```

1203 --      copyright          = "Hossein Movahhedian",
1204 --      license              = "LPPL v1.3c"
1205 -- }
1206 --
1207 -- -- ^A%% texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
1208 -- local err, warn, info, log = luatexbase.provides_module(texnegar luatex kashida.module)
1209 -- texnegar luatex kashida.log      = log or (function (s) luatexbase.module_info("texnegar
1210 -- texnegar luatex kashida.warning = warn or (function (s) luatexbase.module_warning("texneg
1211 -- texnegar luatex kashida.error   = err or (function (s) luatexbase.module_error("texnegar
1212
1213 local peCharTableInitial, peCharTableMedial, peCharTableFinal = dofile(kpse.find_file("texnegar
char-table.lua"))
1214
1215 local kashida_unicode = 1600
1216 local kashida_subtype = 256
1217
1218 local COLORSTACK = node.subtype("pdf_colorstack")
1219 local node_id     = node.id
1220 local GLUE        = node_id("glue")
1221 local GLYPH       = node_id("glyph")
1222 local HLIST       = node_id("hlist")
1223 local RULE        = node_id("rule")
1224 local WHATSIT     = node_id("whatsit")
1225
1226 local l_texnegar_kashida_glyph_bool      = token.create("l_texnegar_kashida_glyph_bool")
1227 local l_texnegar_kashida_leaders_glyph_bool = token.create("l_texnegar_kashida_leaders_glyph
1228 local l_texnegar_kashida_leaders_hrule_bool = token.create("l_texnegar_kashida_leaders_hrule
1229
1230 local l_texnegar_hboxrecursion_bool      = token.create("l_texnegar_hboxrecursion_bool")
1231 local l_texnegar_vboxrecursion_bool      = token.create("l_texnegar_vboxrecursion_bool")
1232
1233 local selected_font = font.current()
1234 local selected_font_old = selected_font
1235
1236 local string_format = string.format
1237 local debug_getinfo = debug.getinfo
1238
1239 function GetGlyphDimensions(font_file, glyph_index)
1240     local funcName     = debug.getinfo(1).name
1241     local funcNparams  = debug.getinfo(1).nparams
1242
1243     local fnt = fontloader.open(font_file)
1244     local idx = 0
1245     local fnt_glyphcnt = fnt.glyphcnt
1246     local fnt_glyphmin = fnt.glyphmin
1247     local fnt_glyphmax = fnt.glyphmax
1248     if fnt_glyphcnt > 0 then
1249         for idx = fnt_glyphmin, fnt_glyphmax do
1250             local gl = fnt.glyphs[idx]
1251             if gl then
1252                 local gl_unicode = gl.unicode
1253                 if gl_unicode == glyph_index then
1254                     local gl_name     = gl.name
1255                     gl_width  = gl.width

```

```

1256         local gl_bbox    = gl.boundingBox
1257         gl_llx            = gl_bbox[1]
1258         gl_depth          = gl_bbox[2]
1259         gl_urx            = gl_bbox[3]
1260         gl_height         = gl_bbox[4]
1261         break
1262     end
1263 end
1264     idx = idx + 1
1265 end
1266 end
1267 fontloader.close(fnt)
1268 return {width = gl_width, height = gl_height, depth = gl_depth, llx = gl_llx, urx = gl_urx}
1269 end
1270
1271 function GetGlue(t_plb_line_glue_node, t_plb_node)
1272     local funcName      = debug_getinfo(1).name
1273     local funcNparams   = debug_getinfo(1).nparams
1274
1275     local glue_id       = t_plb_line_glue_node.id
1276     local glue_subtype  = t_plb_line_glue_node.subtype
1277     local glue_width    = t_plb_line_glue_node.width
1278     local glue_stretch  = t_plb_line_glue_node.stretch
1279     local glue_shrink   = t_plb_line_glue_node.shrink
1280     local eff_glue_width = node.effective_glue(t_plb_line_glue_node, t_plb_node)
1281     local glue_stretch_order = t_plb_line_glue_node.stretch_order
1282     local glue_shrink_order = t_plb_line_glue_node.shrink_order
1283     local glue_delta    = 0
1284     glue_delta = eff_glue_width - glue_width
1285     return { id = glue_id, subtype = glue_subtype, width = glue_width, stretch = glue_stretch,
1286             shrink = glue_shrink, stretch_order = glue_stretch_order, shrink_order = glue_shrink_order,
1287             effective_glue = eff_glue_width, delta = glue_delta }
1288 end
1289
1290 function GetGlyph(t_plb_line_glyph_node, t_tbl_line_fields, t_CharTableInitial, t_CharTableMedial, t_CharTableFinal)
1291     local funcName      = debug_getinfo(1).name
1292     local funcNparams   = debug_getinfo(1).nparams
1293
1294     local glyph_id      = t_plb_line_glyph_node.id
1295     local glyph_subtype = t_plb_line_glyph_node.subtype
1296     local glyph_char    = t_plb_line_glyph_node.char
1297     local glyph_font    = t_plb_line_glyph_node.font
1298     local glyph_lang    = t_plb_line_glyph_node.lang
1299     local glyph_width   = t_plb_line_glyph_node.width
1300     local glyph_data    = t_plb_line_glyph_node.data
1301     if not (t_CharTableInitial[glyph_char] == nil) then
1302         t_tbl_line_fields.joinerCharInitial = t_tbl_line_fields.joinerCharInitial + 1
1303         t_plb_line_glyph_node.data = 1
1304     elseif not (t_CharTableMedial[glyph_char] == nil) then
1305         t_tbl_line_fields.joinerCharMedial = t_tbl_line_fields.joinerCharMedial + 1
1306         t_plb_line_glyph_node.data = 2
1307     elseif not (t_CharTableFinal[glyph_char] == nil) then
1308         t_tbl_line_fields.joinerCharFinal = t_tbl_line_fields.joinerCharFinal + 1
1309         t_plb_line_glyph_node.data = 3
1310     end
1311 end

```

```

1310     end
1311     return { id = glyph_id, subtype = glyph_subtype, char = glyph_char, font = glyph_font, l
1312 end
1313
1314 function ProcessTableKashidaHlist(ksh_hlistNode, hbox_num, in_font)
1315     local funcName      = debug_getinfo(1).name
1316     local funcNparams = debug_getinfo(1).nparams
1317
1318     local ksh_hlistNode_id      = ksh_hlistNode.id
1319     local ksh_hlistNode_subtype = ksh_hlistNode.subtype
1320
1321     for tn in node.traverse(ksh_hlistNode.head) do
1322         local tn_id = tn.id
1323         local tn_subtype = tn.subtype
1324
1325         if tn_id == 0 then
1326             for tp in node.traverse(tn.head) do
1327                 local tp_id = tp.id
1328                 local tp_subtype = tp.subtype
1329                 if tp_id == 29 then
1330                     if l_texnegar_color_bool.mode == c_true_bool.mode then
1331                         local col_str      = color_tbl[1] .. " " .. color_tbl[2] .. " " .. c
1332                         local col_str_rg   = col_str .. " rg "
1333                         local col_str_RG  = col_str .. " RG"
1334
1335                         local color_push   = node.new(WHATSIT, COLORSTACK)
1336                         local color_pop    = node.new(WHATSIT, COLORSTACK)
1337                         color_push.stack   = 0
1338                         color_pop.stack    = 0
1339                         color_push.command = 1
1340                         color_pop.command  = 2
1341                         glue_ratio        = .2
1342                         color_push.data    = col_str_rg .. col_str_RG
1343                         color_pop.data     = col_str_rg .. col_str_RG
1344                         tn.head = node.insert_before(tn.list, tn.head, node.copy(color_push))
1345                         tn.head = node.insert_after(tn.list, node.tail(tn.head), node.copy(c
1346                     end
1347
1348                     local tp_font = tp.font
1349                     local tp_char = tp.char
1350                     tp.font = in_font
1351
1352                     local ksh_unicode
1353                     ksh_unicode = font.getfont(in_font).resources.unicodes['kashida']
1354                     if hbox_num == 'l_texnegar_k_box' then
1355                         tp.char = current_kashida_unicode or kashida_unicode
1356                     elseif hbox_num == 'l_texnegar_ksh_box' then
1357                         tp.char = ksh_unicode
1358                         tn_width = tn.width
1359                         ksh_hlistNode.width = tn_width
1360                     end
1361                 elseif tp_id == 0 then
1362                     if tp_subtype ~= 3 then
1363                         tbl_kashida_hlist_nodes[ #tbl_kashida_hlist_nodes + 1 ] = tp

```

```

1364         end
1365     end
1366 end
1367 elseif tn_id == 1 then
1368     do end
1369 elseif tn_id == 8 then
1370     do end
1371 elseif tn_id == 29 then
1372     if l_texnegar_color_bool.mode == c_true_bool.mode then
1373         local col_str      = color_tbl[1] .. " " .. color_tbl[2] .. " " .. color_tbl[3]
1374         local col_str_rg   = col_str .. " rg "
1375         local col_str_RG   = col_str .. " RG"
1376
1377         local color_push   = node.new(WHATSIT, COLORSTACK)
1378         local color_pop    = node.new(WHATSIT, COLORSTACK)
1379         color_push.stack   = 0
1380         color_pop.stack    = 0
1381         color_push.command = 1
1382         color_pop.command  = 2
1383         glue_ratio        = .2
1384         color_push.data    = col_str_rg .. col_str_RG
1385         color_pop.data     = col_str_rg .. col_str_RG
1386         ksh_hlistNode.head = node.insert_before(ksh_hlistNode.list, ksh_hlistNode.head, color_push)
1387         ksh_hlistNode.head = node.insert_after(ksh_hlistNode.list, node.tail(ksh_hlistNode.list), color_pop)
1388     end
1389
1390     local tn_font = tn.font
1391     local tn_char = tn.char
1392     tn.font = in_font
1393
1394     local ksh_unicode
1395     ksh_unicode = font.getfont(in_font).resources.unicodes['kashida']
1396     if hbox_num == 'l_texnegar_k_box' then
1397         tn.char = kashida_unicode
1398     elseif hbox_num == 'l_texnegar_ksh_box' then
1399         tn.char = ksh_unicode
1400         tn_width = tn.width
1401         ksh_hlistNode.width = tn_width
1402     end
1403 else
1404     print(string_format("\n tn. Not processed node id is: %d", tn_id))
1405 end
1406 end
1407 end
1408
1409 function SetFontInHbox(hbox_num, font_num)
1410     local funcName = debug_getinfo(1).name
1411     local funcNparams = debug_getinfo(1).nparams
1412
1413     tbl_kashida_hlist_nodes = {}
1414
1415     local tmp_node
1416     tmp_node = node.new("hlist")
1417     tmp_node = tex.getbox(hbox_num)

```

```

1418
1419     ProcessTableKashidaHlist(tmp_node, hbox_num, font_num)
1420
1421     ::kashida_hlist_BEGIN::
1422     if #tbl_kashida_hlist_nodes > 0 then
1423         local kashida_hlistNodeAdded = table.remove(tbl_kashida_hlist_nodes,1)
1424         ProcessTableKashidaHlist(kashida_hlistNodeAdded, hbox_num, font_num)
1425         goto kashida_hlist_BEGIN
1426     end
1427 end
1428
1429 function StretchGlyph(t_plb_node, t_plb_glyph_node, t_gluePerJoiner, t_dir, t_filler)
1430     local funcName      = debug_getinfo(1).name
1431     local funcNparams    = debug_getinfo(1).nparams
1432
1433     if t_filler == "resized_kashida" then
1434         SetFontInHbox('l_texnegar_k_box', selected_font)
1435     elseif t_filler == "leaders+kashida" then
1436         SetFontInHbox('l_texnegar_ksh_box', selected_font)
1437     end
1438
1439     kashida_node = node.new(GLYPH)
1440     node_glue     = node.new(GLUE)
1441     node_rule     = node.new(RULE)
1442     node_hlist    = node.new(HLIST)
1443
1444     font_current = selected_font
1445     font_name    = font.fonts[font_current].fullname
1446     font_file    = font.fonts[font_current].filename
1447     kashida_char = font.fonts[font_current].characters[1600]
1448
1449     kashida_node.subtype = kashida_subtype
1450     kashida_node.font    = font_current
1451     if string.match(font_name, "^(Amiri).*" ) == "Amiri" then
1452         kashida_node.char = current_kashida_unicode
1453     else
1454         kashida_node.char = kashida_unicode
1455     end
1456     kashida_node.lang    = tex.language
1457
1458     kashida_width  = kashida_node.width
1459     kashida_height = kashida_node.height
1460     kashida_depth  = kashida_node.depth
1461
1462     tbl_gl_dimen = GetGlyphDimensions(font_file, kashida_unicode)
1463     ksh_width, ksh_height, ksh_depth, ksh_llx, ksh_urx =
1464         tbl_gl_dimen.width, tbl_gl_dimen.height, tbl_gl_dimen.depth, tbl_gl_dimen.llx, tbl_g
1465
1466     ratio_width = kashida_width / ksh_width
1467     leaders_height = ratio_width * ksh_height
1468     leaders_depth = - ratio_width * ksh_depth
1469
1470     node_glue.subtype = 100
1471     node.setglue(node_glue, t_gluePerJoiner, 0, 0, 0, 0)

```

```

1472
1473 if t_filler == "resized_kashida" then
1474     node_glue.leader = node.copy_list(tex.box['l_texnegar_k_box'])
1475 elseif t_filler == "leaders+kashida" then
1476     node_glue.leader = node.copy_list(tex.box['l_texnegar_ksh_box'])
1477 elseif t_filler == "leaders+hrule" then
1478     node_glue.leader = node_rule
1479 end
1480
1481 node_glue.leader.subtype = 0
1482 node_glue.leader.height = leaders_height
1483 node_glue.leader.depth = leaders_depth
1484
1485 node_glue.leader.dir = t_dir
1486
1487 node.insert_after(t_plb_node.list, t_plb_glyph_node, node_glue)
1488 if t_filler == "leaders+hrule" then
1489     for tn in node.traverse(t_plb_node.head) do
1490         local tn_id = tn.id
1491         local tn_subtype = tn.subtype
1492
1493         if tn_id == 12 and tn_subtype == 100 then
1494             local t_hbox = node.new(HLIST)
1495             local t_hrule = node.copy(tn)
1496
1497             if string.match(font_name, "^(Amiri).*") == "Amiri" then
1498                 t_hrule.leader.height = kashida_height
1499                 t_hrule.leader.depth = kashida_depth
1500             end
1501
1502             t_hbox.head = node.insert_after(t_hbox.list, t_hbox.head, t_hrule)
1503             t_plb_node.head = node.insert_after(t_plb_node.list, tn, t_hbox)
1504
1505             if l_texnegar_color_bool.mode == c_true_bool.mode then
1506                 local col_str = color_tbl[1] .. " " .. color_tbl[2] .. " " .. color
1507                 local col_str_rg = col_str .. " rg "
1508                 local col_str_RG = col_str .. " RG"
1509
1510                 local color_push = node.new(WHATSIT, COLORSTACK)
1511                 local color_pop = node.new(WHATSIT, COLORSTACK)
1512                 color_push.stack = 0
1513                 color_pop.stack = 0
1514                 color_push.command = 1
1515                 color_pop.command = 2
1516                 glue_ratio = .2
1517                 color_push.data = col_str_rg .. col_str_RG
1518                 color_pop.data = col_str_rg .. col_str_RG
1519                 t_hbox.head = node.insert_before(t_hbox.list, t_hbox.head, node.copy(col
1520                 t_hbox.head = node.insert_after(t_hbox.list, node.tail(t_hbox.head), nod
1521             end
1522         end
1523     end
1524 end
1525 end

```

```

1526
1527 function GetFillerSpec(t_plb_node, t_plb_head_node, t_tbl_line_fields, t_CharTableInitial, t
1528     local funcName      = debug_getinfo(1).name
1529     local funcNparams = debug_getinfo(1).nparams
1530
1531     t_plb_node_id = t_plb_node.id
1532     t_plb_node_subtype = t_plb_node.subtype
1533
1534     for p in node.traverse(t_plb_head_node) do
1535         local p_id = p.id
1536         local p_subtype = p.subtype
1537         if p_id == 0 then
1538             t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - p.
1539             if p_subtype ~= 3 then
1540                 tbl_hlist_nodes[ #tbl_hlist_nodes + 1 ] = p
1541             end
1542         elseif p_id == 1 then
1543             t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - p.
1544             tbl_vlist_nodes[ #tbl_vlist_nodes + 1 ] = p
1545         elseif p_id == 12 then
1546             tbl_p_glue = GetGlue(p, t_plb_node)
1547             t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - tb
1548             t_tbl_line_fields.total_glues = t_tbl_line_fields.total_glues + 1
1549             t_tbl_line_fields.stretchedGlue = t_tbl_line_fields.stretchedGlue + tbl_p_glue["
1550         elseif p_id == 29 then
1551             tbl_p_glyph, t_tbl_line_fields = GetGlyph(p, t_tbl_line_fields, t_CharTableIniti
1552             selected_font_old = selected_font
1553             selected_font = tbl_p_glyph["font"]
1554             t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - tb
1555             t_tbl_line_fields.total_glyphs = t_tbl_line_fields.total_glyphs + 1
1556         end
1557     end
1558
1559     t_tbl_line_fields.total_joiners = t_tbl_line_fields.joinerCharInitial + t_tbl_line_field
1560     t_tbl_line_fields.gluePerJoiner = 0
1561     if t_tbl_line_fields.total_glues == 0 then
1562         t_tbl_line_fields.stretchedGlue = t_tbl_line_fields.lineWidthRemainder
1563     end
1564     if t_tbl_line_fields.total_joiners > 0 then
1565         t_tbl_line_fields.gluePerJoiner = t_tbl_line_fields.stretchedGlue // t_tbl
1566         t_tbl_line_fields.stretchedGlueRemainder = t_tbl_line_fields.stretchedGlue % t_tbl
1567     elseif t_tbl_line_fields.total_joiners == 1 then
1568         t_tbl_line_fields.gluePerJoiner = t_tbl_line_fields.stretchedGlue
1569     end
1570
1571     return t_tbl_line_fields
1572 end
1573
1574 function ProcessTableHlist(tmphl_n)
1575     local funcName      = debug_getinfo(1).name
1576     local funcNparams = debug_getinfo(1).nparams
1577
1578     local tmphl_n_id      = tmphl_n.id
1579     local tmphl_n_subtype = tmphl_n.subtype

```

```

1580
1581     local tbl_line_fields = { line_dir          = "", line_width          = 0, lineWidthRemaind
1582                               joinerCharInitial = 0, joinerCharMedial = 0, joinerCharFinal
1583                               stretchedGlue     = 0, total_glues        = 0, gluePerJoiner
1584
1585     local tbl_p_glue, tbl_p_glyph
1586
1587     if (tmphl_n_id == 0) and (tmphl_n_subtype == 1 or tmphl_n_subtype == 2) then
1588         tbl_line_fields.line_width = tmphl_n.width
1589         tbl_line_fields.line_dir   = tmphl_n.dir
1590         tbl_line_fields.lineWidthRemainder = tbl_line_fields.line_width
1591
1592         if tbl_line_fields.line_dir == "TLT" then
1593             tbl_line_fields = GetFillerSpec(tmphl_n, tmphl_n.head, tbl_line_fields, peCharTa
1594
1595             if tbl_line_fields.total_joiners == 0 or tbl_line_fields.gluePerJoiner == 0 or
1596                 goto continue
1597             end
1598
1599             for q in node.traverse_id(GLUE, tmphl_n.head) do
1600                 local eff_glue_width = node.effective_glue(q, tmphl_n)
1601                 node.setglue(q, q.width, 0, 0, q.stretch_order, q.glue_shrink_order)
1602             end
1603
1604             for r in node.traverse_id(GLYPH, tmphl_n.head) do
1605                 local r_data = r.data
1606                 if r_data == 1 or r_data == 2 then
1607                     StretchGlyph(tmphl_n, r, tbl_line_fields.gluePerJoiner, tbl_line_fields.
1608                 elseif r_data == 3 then
1609                     goto for_loop_01
1610                 end
1611                 ::for_loop_01::
1612             end
1613             tbl_line_fields.line_width = tmphl_n.width
1614             tbl_line_fields.lineWidthRemainder = line_width
1615         elseif tbl_line_fields.line_dir == "TRT" then
1616             tbl_line_fields = GetFillerSpec(tmphl_n, tmphl_n.head, tbl_line_fields, peCharTa
1617             if tbl_line_fields.total_joiners == 0 or tbl_line_fields.gluePerJoiner == 0 or
1618                 goto continue
1619             end
1620
1621             for q in node.traverse_id(GLUE, tmphl_n.head) do
1622                 local eff_glue_width = node.effective_glue(q, tmphl_n)
1623                 node.setglue(q, q.width, 0, 0, q.stretch_order, q.glue_shrink_order)
1624             end
1625
1626             for r in node.traverse_id(GLYPH, tmphl_n.head) do
1627                 local r_data = r.data
1628                 if r_data == 1 or r_data == 2 then
1629                     StretchGlyph(tmphl_n, r, tbl_line_fields.gluePerJoiner, tbl_line_fields.
1630                 elseif r_data == 3 then
1631                     goto for_loop_02
1632                 end
1633                 ::for_loop_02::

```

```

1634         end
1635         tbl_line_fields.line_width = tmphl_n.width
1636         tbl_line_fields.lineWidthRemainder = line_width
1637     else
1638         print(string_format("\n Line direction '%s' is not supported yet!", tbl_line_fie
1639     end
1640 end
1641 ::continue::
1642 end
1643
1644 function ProcessTableVlist(tmpvl_n)
1645     local funcName      = debug_getinfo(1).name
1646     local funcNparams   = debug_getinfo(1).nparams
1647
1648     local tmpvl_n_id     = tmpvl_n.id
1649     local tmpvl_n_subtype = tmpvl_n.subtype
1650
1651     print(string_format(" %s: 00-0 tmpvl_n: id: %d, subtype: %d", funcName, tmpvl_n_id, tmpv
1652     for vbNode in node.traverse(tmpvl_n) do
1653         if vbNode.id == 1 and vbNode.subtype == 0 then
1654             for tr_vbNode in node.traverse(vbNode.head) do
1655                 if (tr_vbNode.id == 0) and (tr_vbNode.subtype == 1 or tr_vbNode.subtype ==
1656                     ProcessTableHlist(tr_vbNode)
1657             end
1658         end
1659     end
1660 end
1661 end
1662
1663 function PostLineBreakFilter(hboxes_stack, groupcode)
1664     local funcName      = debug_getinfo(1).name
1665     local funcNparams   = debug_getinfo(1).nparams
1666
1667     funcName = "PostLineBreakFilter"
1668
1669     local tbl_fonts_used = { }
1670     local tbl_fonts_chars = { }
1671     local tbl_fonts_chars_init = { }
1672     local tbl_fonts_chars_medi = { }
1673     local tbl_fonts_chars_fina = { }
1674     tbl_fonts_used, tbl_fonts_chars, tbl_fonts_chars_init, tbl_fonts_chars_medi, tbl_fonts_c
1675
1676     for k1, v1 in pairs(tbl_fonts_chars_init) do
1677         for k2, v2 in pairs(tbl_fonts_chars_init[k1]) do
1678             if k2 and not peCharTableInitial[k2] then
1679                 peCharTableInitial[k2] = utf8.char(k2)
1680             end
1681         end
1682     end
1683
1684     for k1, v1 in pairs(tbl_fonts_chars_medi) do
1685         for k2, v2 in pairs(tbl_fonts_chars_medi[k1]) do
1686             if k2 and not peCharTableMedial[k2] then
1687                 peCharTableMedial[k2] = utf8.char(k2)

```

```

1688         end
1689     end
1690 end
1691
1692 for k1, v1 in pairs(tbl_fonts_chars_fina) do
1693     for k2, v2 in pairs(tbl_fonts_chars_fina[k1]) do
1694         if k2 and not peCharTableFinal[k2] then
1695             peCharTableFinal[k2] = utf8.char(k2)
1696         end
1697     end
1698 end
1699
1700 tbl_hlist_nodes = {}
1701 tbl_vlist_nodes = {}
1702 for hlistNode in node.traverse(hboxes_stack) do
1703     if node.next(hlistNode) == nil then
1704         goto END
1705     end
1706
1707     ProcessTableHlist(hlistNode)
1708
1709     if l_texpnegar_hboxrecursion_bool.mode == c_true_bool.mode then
1710         ::hboxBEGIN::
1711         if #tbl_hlist_nodes > 0 then
1712             local hlistNodeAdded = table.remove(tbl_hlist_nodes,1)
1713             ProcessTableHlist(hlistNodeAdded)
1714             goto hboxBEGIN
1715         end
1716     end
1717
1718     if l_texpnegar_vboxrecursion_bool.mode == c_true_bool.mode then
1719         ::vboxBEGIN::
1720         if #tbl_vlist_nodes > 0 then
1721             local vlistNodeAdded = table.remove(tbl_vlist_nodes,1)
1722             ProcessTableVlist(vlistNodeAdded)
1723             goto vboxBEGIN
1724         end
1725     end
1726
1727     ::END::
1728 end
1729 return hboxes_stack
1730 end
1731
1732 if l_texpnegar_kashida_glyph_bool.mode == c_true_bool.mode then
1733     filler_pe = "resized_kashida"
1734 elseif l_texpnegar_kashida_leaders_glyph_bool.mode == c_true_bool.mode then
1735     filler_pe = "leaders+kashida"
1736 elseif l_texpnegar_kashida_leaders_hrulerule_bool.mode == c_true_bool.mode then
1737     filler_pe = "leaders+hrulerule"
1738 else
1739     print(string_format" Unknown kashida value.")
1740 end
1741

```

```

1742 function StartStretching()
1743     if not luatexbase.in_callback('post_linebreak_filter', 'insertKashida') then
1744         luatexbase.add_to_callback('post_linebreak_filter', PostLineBreakFilter, 'insertKashida')
1745     end
1746 end
1747
1748 function StopStretching()
1749     if luatexbase.in_callback('post_linebreak_filter', 'insertKashida') then
1750         luatexbase.remove_from_callback('post_linebreak_filter', 'insertKashida')
1751     end
1752 end
1753 --
1754 --
1755 -- End of file 'texnegar-luatex-kashida.lua'.
1756 </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_EX book*, Addison-Wesley, 1986.

- [2] Victor Eijkhout, *TeX BY TOPIC*, Addison-Wesley, 2013.
- [3] Paul W. Abrahams, Kathryn A. Hargreaves, and Karl Berry, *TeX for the Impatient*, Addison-Wesley, 2013.
- [4] Leslie Lamport, *LaTeX, A document preparation System*, Addison-Wesley, 1986.
- [5] Frank Mittelbach and Michel Goossens with Johannes Braams, David Carlisle, and Chris Rowley, *The LaTeX 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 LaTeX 2_ε Sources*, [CTAN:macros/latex/base/source2e.pdf](#), 2020-02-02
- [9] Package `l3kernel`: The LaTeX3 Team, *The LaTeX3 Sources*, [CTAN:macros/latex/contrib/l3kernel/source3.pdf](#), 2020-07-17
- [10] Package `l3kernel`: The LaTeX3 Team, *The LaTeX3 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_YTeX reference guide, [CTAN:info/xetexref/xetex-reference.pdf](#), 2019-12-09
- [13] Package `xetex`: Jonathan Kew, About X_YTeX, [CTAN:systems/doc/xetex/XeTeX-notes.pdf](#), 2005-10-17
- [14] Package `xetex`: Michel Goossens, The X_YTeX Companion, <http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf>, 2009-08-19
- [15] Website: Stack Exchange: Hot Questions, T_EX-L_AT_EX Q&A for users of TeX, LaTeX, ConTeXt, and related typesetting systems, tex.stackexchange.com
- [16] Website: LuaTeX Wiki, LuaT_EX Wiki, 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.

Symbols		<code>\2</code> 648
<code>\[</code>	648	
<code>\]</code>	25	
<code>_</code>	648	
		<code>_</code> 521, 562, 648
Numbers		
<code>\0</code>	562	A
<code>\1</code>	297, 648	<code>\AtBeginDocument</code> 55, 293

B		F	
bool commands:		fi commands:	
\bool_if:N	38, 51, 69, 474, 490, 500, 525, 582, 596, 641, 715, 720, 741, 746	\fi:	42, 547, 635
\bool_if:nTF	511, 651, 735	G	
\bool_set_false:N	131, 135, 137, 138, 139, 141, 142, 143, 144, 145, 275, 317, 429, 435, 446, 452, 468	\gdef	610, 620
\bool_set_true:N	279, 299, 300, 305, 306, 311, 312, 322, 323, 340, 397, 411, 433, 437, 450, 454, 467	\GenericError	530
box commands:		H	
\box_new:N	81, 82	hbox commands:	
C		\hbox_set:Nn	44, 46, 653
\c	562	\height	44, 634
\char	46, 521	I	
clist commands:		if commands:	
\clist_count:N	504	\if_int_compare:w	40
\clist_item:Nn	509, 510	\if_meaning:w	625
\clist_map_inline:Nn	516, 684, 690, 696, 702, 708	\if_mode_vertical:	529
\clist_new:N	502	\IfNoValueF	552, 554
\clist_put_left:Nn	513	\input	492
\clist_set:Nn	208, 209, 210, 211, 212, 213, 214, 215, 216, 218, 510, 683, 689, 695, 701, 707	int commands:	
\l_tmpa_clist	510, 513	\l_fontnumber_int	109
\color	589	\int_case:nnTF	332
\colorlet	584, 587	\int_const:Nn	87, 88, 89, 91, 92
\convertcolorspec	414	\int_eval:n	508
\copy	655	\int_new:N	97, 99, 101, 103, 104, 105, 106, 107, 109, 503
cs commands:		\int_set:Nn	147, 148, 149, 150, 151, 331, 334, 335, 336, 337, 338, 339, 504, 507, 508, 553
\cs:w	610, 620, 626, 630	\int_step_inline:nnnn	505
\cs_end:	610, 620, 626, 630	\int_use:N	673, 674
\cs_new:Nn	528	\c_one_int	600
\cs_new:Npn	594, 639	\l_tmpa_int	331, 332, 507, 508, 509
\cs_new_protected:Nn	559	\l_tmpb_int	508, 510
D		iow commands:	
\def	79	\iow_now:Nn	608, 618
dim commands:		K	
\dim_compare:nTF	545, 632	\KashidaHMFixOff	461, 468, 472
\dim_eval:n	630	\KashidaHMFixOn	462, 467, 471
\dim_new:N	129	\KashidaOff	472
\dim_set:Nn	628	\KashidaOn	57, 471
\directlua	48, 417, 461, 462	keys commands:	
\discouragebadlinebreaks	550	\keys_define:nn	266
E		L	
else commands:		\llap	634
\else:	541, 627	lua commands:	
\endinput	16, 21, 31, 61, 74, 495, 567, 765	\lua_now:n	46
exp commands:		\luatexversion	40
\exp_after:wN	610, 620, 624	M	
		\makeatletter	482
		\makeatother	488
		\MessageBreak	537, 539

<code>\l_texnegar_fnt_roya_tl</code>	174, 357	<code>\l_texnegar_kashida_slot_int</code> . . .	99
<code>\l_texnegar_fnt_shafigh_tl</code> .	175, 358	<code>\l_texnegar_ksh_box</code> .	46, 82, 653, 655
<code>\l_texnegar_fnt_shafighKurd_tl</code> . .		<code>\c_texnegar_ksh_int</code> .	87, 634, 673, 674
	176, 359	<code>\c_texnegar_l_charclass</code>	
<code>\l_texnegar_fnt_shafighUzbek_tl</code> .			573, 698, 726, 727,
	177, 360		728, 731, 732, 752, 753, 754, 757, 758
<code>\l_texnegar_fnt_shiraz_tl</code> . .	178, 361	<code>\l_texnegar_l_clist</code>	695, 696
<code>\l_texnegar_fnt_sols_tl</code>	179, 362	<code>\l_texnegar_lig_aalt_clist</code> .	208, 220
<code>\l_texnegar_fnt_tabriz_tl</code> . .	180, 363	<code>\l_texnegar_lig_aalt_tl</code>	196, 220, 387
<code>\l_texnegar_fnt_titr_tl</code>	181, 364	<code>\l_texnegar_lig_ccmp_clist</code> .	209, 221
<code>\l_texnegar_fnt_titre_tl</code> . . .	182, 365	<code>\l_texnegar_lig_ccmp_tl</code>	197, 221, 388
<code>\l_texnegar_fnt_traffic_tl</code> .	183, 366	<code>\l_texnegar_lig_default_clist</code> . .	216
<code>\l_texnegar_fnt_vahid_tl</code> . . .	184, 367	<code>\l_texnegar_lig_default_tl</code>	
<code>\l_texnegar_fnt_vosta_tl</code> . . .	185, 368		204, 395, 399, 511
<code>\l_texnegar_fnt_yaghut_tl</code> . .	186, 369	<code>\l_texnegar_lig_dlig_clist</code> .	210, 222
<code>\l_texnegar_fnt_yagut_tl</code> . . .	187, 370	<code>\l_texnegar_lig_dlig_tl</code>	198, 222, 389
<code>\l_texnegar_fnt_yas_tl</code>	188, 371	<code>\l_texnegar_lig_fina_clist</code> .	211, 223
<code>\l_texnegar_fnt_yekan_tl</code> . . .	189, 372	<code>\l_texnegar_lig_fina_tl</code>	199, 223, 390
<code>\l_texnegar_fnt_yermook_tl</code> .	190, 373	<code>\l_texnegar_lig_init_clist</code> .	212, 224
<code>\l_texnegar_fnt_zar_tl</code>	191, 374	<code>\l_texnegar_lig_init_tl</code>	200, 224, 391
<code>\l_texnegar_fnt_ziba_tl</code>	192, 375	<code>\l_texnegar_lig_locl_clist</code> .	213, 225
<code>\l_texnegar_font_full_tl</code>	116, 645, 646	<code>\l_texnegar_lig_locl_tl</code>	201, 225, 392
<code>\l_texnegar_font_init_tl</code>	647, 648, 651	<code>\l_texnegar_lig_medi_clist</code> .	214, 226
<code>\l_texnegar_font_name_tl</code>		<code>\l_texnegar_lig_medi_tl</code>	202, 226, 393
	117, 646, 647, 659	<code>\l_texnegar_lig_names_clist</code>	
<code>\c_texnegar_four_int</code>	92, 673		218, 504, 509, 510
<code>\l_texnegar_gap_filler_tl</code>		<code>\l_texnegar_lig_names_len_int</code> . . .	
	123, 292, 304, 310, 316,		503, 504, 505
	321, 325, 326, 643, 713, 736, 737, 761	<code>\l_texnegar_lig_rlig_clist</code> .	215, 227
<code>\l_texnegar_hboxrecursion_bool</code> . .		<code>\l_texnegar_lig_rlig_tl</code>	203, 227, 394
	143, 429, 433, 435, 437	<code>\l_texnegar_ligature_bool</code>	
<code>\l_texnegar_hboxrecursion_off_tl</code>			141, 397, 500
	159, 427	<code>\l_texnegar_ligatures_clist</code>	
<code>\l_texnegar_hboxrecursion_on_tl</code> .			502, 513, 516
	160, 431	<code>\texnegar_line_break:</code>	528
<code>\l_texnegar_high_penalty_int</code> . . .		<code>\l_texnegar_line_break_penalty_-</code>	
	106, 150, 337	<code>int</code>	101, 334,
<code>\l_texnegar_k_box</code>	44, 81		335, 336, 337, 338, 339, 544, 550, 553
<code>\l_texnegar_kashida_fix_bool</code> . . .		<code>\l_texnegar_line_break_tl</code>	
	38, 51, 69,		111, 561, 562, 563
	135, 299, 305, 311, 317, 322, 467,	<code>\l_texnegar_linebreakpenalty_-</code>	
	468, 474, 596, 641, 715, 720, 741, 746	<code>bool</code>	142, 340, 525
<code>\texnegar_kashida_glyph</code>		<code>\l_texnegar_low_penalty_int</code>	
	594, 716, 717, 721, 722,		104, 148, 335
	724, 725, 726, 727, 728, 729, 730, 731	<code>\c_texnegar_lrm_int</code>	88, 598, 670
<code>\l_texnegar_kashida_glyph_bool</code> . .		<code>\c_texnegar luatexversionmajormin_-</code>	
	137, 300	<code>int</code>	40, 41, 84
<code>\texnegar_kashida_leaders</code>		<code>\c_texnegar luatexversionminormin_-</code>	
	639, 742, 743, 747, 748,	<code>int</code>	40, 41, 85
	750, 751, 752, 753, 754, 755, 756, 757	<code>\l_texnegar_main_font_full_tl</code> . . .	
<code>\l_texnegar_kashida_leaders_-</code>			113, 295, 296
<code>glyph_bool</code>	138, 306, 323	<code>\l_texnegar_main_font_name_tl</code> . . .	
<code>\l_texnegar_kashida_leaders_-</code>			114, 296, 297
<code>hrule_bool</code>	139, 312		

<code>\l_texnegar_max_penalty_int</code>	tl commands:
107, 151, 338	<code>\tl_case:Nn</code> 271
<code>\l_texnegar_med_penalty_int</code>	<code>\tl_case:NnTF</code> 287, 346, 385, 425, 442
105, 149, 336	<code>\tl_const:Nn</code> 84, 85, 94, 95, 521
<code>\l_texnegar_min_penalty_int</code>	<code>\tl_if_empty:NnTF</code> 326, 404, 476
103, 147, 334	<code>\tl_if_eq:NnTF</code> 643, 713
<code>\l_texnegar_minimal_bool</code>	<code>\tl_if_eq_p:Nn</code> 511, 736, 737
131, 275, 279, 490	<code>\tl_new:N</code> 111, 113, 114,
<code>\l_texnegar_minimal_off_tl</code> 132, 273	116, 117, 119, 121, 123, 125, 126, 127
<code>\l_texnegar_minimal_on_tl</code> 133, 277	<code>\tl_set:Nn</code>
<code>\l_texnegar_pos_tl</code> 132, 133, 153, 154, 155, 156,
602, 610, 620, 626, 630	157, 159, 160, 162, 163, 165, 166,
<code>\texnegar_put_line_breaks:n</code> 556, 559	167, 168, 169, 170, 171, 172, 173,
<code>\c_texnegar_r_charclass</code>	174, 175, 176, 177, 178, 179, 180,
574, 704, 729, 731, 755, 757	181, 182, 183, 184, 185, 186, 187,
<code>\l_texnegar_r_clist</code> 701, 702	188, 189, 190, 191, 192, 193, 194,
<code>\c_texnegar_skip_a_tl</code>	196, 197, 198, 199, 200, 201, 202,
. 94, 479, 717, 722, 724, 725, 726,	203, 204, 206, 270, 286, 292, 295,
727, 728, 729, 730, 731, 743, 748,	296, 304, 310, 316, 321, 325, 345,
750, 751, 752, 753, 754, 755, 756, 757	348, 349, 350, 351, 352, 353, 354,
<code>\c_texnegar_skip_b_tl</code> 95, 550	355, 356, 357, 358, 359, 360, 361,
<code>\l_texnegar_skip_default_tl</code>	362, 363, 364, 365, 366, 367, 368,
. 119, 348, 349, 350,	369, 370, 371, 372, 373, 374, 375,
351, 352, 353, 354, 355, 356, 357,	376, 377, 378, 380, 384, 387, 388,
358, 359, 360, 361, 362, 363, 364,	389, 390, 391, 392, 393, 394, 395,
365, 366, 367, 368, 369, 370, 371,	396, 399, 403, 406, 409, 424, 441,
372, 373, 374, 375, 376, 377, 378,	476, 479, 509, 555, 561, 580, 602,
380, 476, 479, 555, 716, 721, 742, 747	603, 606, 616, 645, 646, 647, 649, 650
<code>\l_texnegar_stretch_glyph_tl</code>	<code>\tl_use:N</code> 521, 563
. 153, 289, 292, 713	<code>\l_tmpa_tl</code> 270, 271, 286,
<code>\l_texnegar_stretch_leaders_</code>	287, 345, 346, 384, 385, 403, 404,
<code>glyph_tl</code> 154, 302, 304, 321, 643, 736	409, 424, 425, 441, 442, 509, 511,
<code>\l_texnegar_stretch_leaders_</code>	519, 521, 606, 613, 616, 623, 649, 651
<code>hrule_tl</code> 155, 308, 310, 737	<code>\l_tmpb_tl</code> 520, 521, 650, 651
<code>\l_texnegar_stretch_off_tl</code>	token commands:
. 156, 314, 316	<code>\token_to_str:N</code> 610, 620
<code>\l_texnegar_stretch_on_tl</code> 157, 319	
<code>\c_texnegar_two_int</code> 91, 672	
<code>\l_texnegar_use_color_tl</code>	
. 125, 580, 634, 653, 671	
<code>\l_texnegar_vboxrecursion_bool</code> . .	
. 144, 446, 450, 452, 454	
<code>\l_texnegar_vboxrecursion_off_tl</code>	
. 162, 444	
<code>\l_texnegar_vboxrecursion_on_tl</code> . .	
. 163, 448	
<code>\c_texnegar_y_charclass</code>	
576, 710, 714, 719, 724, 740, 745, 750	
<code>\l_texnegar_y_clist</code> 707, 708	
<code>\l_texnegar_zref_tl</code>	
. 603, 605, 610, 615, 620	
<code>\c_texnegar_zwj_int</code> 89, 654,	
656, 670, 676, 716, 717, 721, 722,	
724, 725, 726, 727, 728, 729, 730, 731	

925, 926, 927, 928, 929, 930, 931,	\XeTeXglyphbounds	672, 673
932, 933, 934, 935, 936, 937, 938,	\XeTeXglyphindex	653
942, 943, 944, 945, 946, 947, 948,	\XeTeXinterchartokenstate	681
949, 950, 951, 952, 953, 954, 955,	\XeTeXinterchartoks	
956, 957, 958, 959, 960, 961, 962,	. 714, 719, 724, 725, 726, 727, 728,	
963, 964, 965, 966, 967, 968, 969,	729, 730, 731, 732, 740, 745, 750,	
970, 971, 972, 973, 974, 975, 976, 977	751, 752, 753, 754, 755, 756, 757, 758	
X		
\XeTeXcharclass		686, 692, 698, 704, 710
\XeTeXcharglyph		673, 674
\XeTeXglyph		653
Z		
\zposx		610, 620
\zsaveposx		605, 615