

The T<sub>E</sub>XNegar package  
Kashida justification in LuaT<sub>E</sub>X and XeT<sub>E</sub>X  
Source code documentation

Hossein Movahhedian\*

Released 2020-08-30 v0.1b

---

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

---

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

## Contents

<b>1</b>	<b>TeXNegar Implementation</b>	<b>3</b>
1.1	File: texnegar.sty . . . . .	3
1.2	File: texnegar-luatex.sty . . . . .	3
1.3	File: texnegar-xetex.sty . . . . .	4
1.4	File: texnegar-ini.tex . . . . .	4
1.5	File: texnegar-common-kashida.tex . . . . .	11
1.6	File: texnegar-xetex-kashida.tex . . . . .	13
1.7	File: texnegar-char-table.lua . . . . .	16
1.8	File: texnegar.lua . . . . .	19
1.9	File: texnegar-ini.lua . . . . .	20
1.10	File: texnegar-luatex-kashida.lua . . . . .	21
<b>2</b>	<b>Acknowledgments</b>	<b>31</b>
<b>3</b>	<b>Change History</b>	<b>31</b>
	[2020-08-29 v0.1a] . . . . .	31
	[2020-08-30 v0.1b] . . . . .	31
	<b>Index</b>	<b>32</b>

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

## 1.1 File: texnegar.sty

```
1 <*texnegar-sty>
2 \RequirePackage{xparse}
3 \RequirePackage{l3keys2e}
4 \RequirePackage{graphicx}[2019-11-30]
5 \RequirePackage[dvipsnames,svgnames,x11names]{xcolor}[2016/05/11]
6 \ProvidesExplPackage {texnegar} {2020-08-30} {0.1b} { Full implementation of kashida feature
7
8 \sys_if_engine_luatex:T
9   {
10     \RequirePackageWithOptions{texnegar-luatex}
11     \endinput
12   }
13 \sys_if_engine_xetex:T
14   {
15     \RequirePackageWithOptions{texnegar-xetex}
16     \endinput
17   }
18 \msg_new:nnn {texnegar} {cannot-use-pdftex}
19   {
20     The~ texnegar~ package~ requires~ either~ XeTeX~ or~ LuaTeX.\\
21     You~ must~ change~ your~ typesetting~ engine~ to,~ e.g.,~
22     "xelatex"~ or~ "lualatex" instead~ of~ "latex"~ or~ "pdflatex".
23   }
24 \msg_fatal:nn {texnegar} {cannot-use-pdftex}
25
26 \endinput
27 </texnegar-sty>
```

## 1.2 File: texnegar-luatex.sty

```
28 <*texnegar-luatex-sty>
29 \ProvidesExplPackage {texnegar-luatex} {2020-08-30} {0.1b} { Full implementation of kashida
30
31 \tex_input:D { texnegar-ini.tex }
32
33 \bool_if:NT \l_texnegar_kashida_fix_bool
34   {
35     \if_int_compare:w \luatexversion < \c_texnegar_luatexversionmajormin_int\c_texnegar_luat
36       \msg_error:nnxxx { texnegar } { luatex-version-is-too-old } { !!!! } { \c_texnegar_l
37     \fi:
38
39     \hbox_set:Nn \l_texnegar_k_box { \resizebox{5000sp}{\height}{-} }
40
41     \hbox_set:Nn \l_texnegar_ksh_box { \char\lua_now:n { tex.sprint(0, font.getfont(font.cur
42
43     \directlua{dofile(kpse.find_file("texnegar.lua"))}
44   }
45
46 \bool_if:NT \l_texnegar_kashida_fix_bool
47   {
48     \tex_input:D { texnegar-common-kashida.tex }
```

```

49
50     \AtBeginDocument
51     {
52         \KashidaOn
53     }
54 }
55
56 \endinput
57 </texnegar-luatex-sty>

```

### 1.3 File: texnegar-xetex.sty

```

58 <*texnegar-xetex-sty>
59 \RequirePackage{zref-savepos}[2020-03-03]
60 \ProvidesExplPackage {texnegar-xetex} {2020-08-30} {0.1b} { Full implementation of kashida f
61
62 \tex_input:D { texnegar-ini.tex }
63
64 \bool_if:NT \l_texnegar_kashida_fix_bool
65 {
66     \tex_input:D { texnegar-xetex-kashida.tex }
67 }
68
69 \endinput
70 </texnegar-xetex-sty>

```

### 1.4 File: texnegar-ini.tex

```

71 <*texnegar-ini-tex>
72 \ProvidesExplFile {texnegar-ini.tex} {2020-08-30} {0.1b} { Full implementation of kashida fe
73
74 \def\TeXNegar{\TeX Negar}
75
76 \box_new:N \l_texnegar_k_box
77 \box_new:N \l_texnegar_ksh_box
78
79 \tl_const:Nn \c_texnegar luatexversionmajormin_int {1}
80 \tl_const:Nn \c_texnegar luatexversionminormin_int {12}
81
82 \int_const:Nn \c_texnegar_ksh_int {"0640} % kashida
83 \int_const:Nn \c_texnegar_lrm_int {"200E} % left-right-mark
84 \int_const:Nn \c_texnegar_zwj_int {"200D} % zero-width joiner
85
86 \int_const:Nn \c_texnegar_two_int {2}
87 \int_const:Nn \c_texnegar_four_int {4}
88
89 \tl_const:Nn \c_texnegar_skip_a_tl { 0 em plus 0.5 em }
90 \tl_const:Nn \c_texnegar_skip_b_tl { 0.14 em plus 5.5 em }
91
92 \int_new:N \l_texnegar_counter_int
93
94 \int_new:N \l_texnegar_kashida_slot_int
95
96 \int_new:N \l_texnegar_line_break_penalty_int
97
98 \int_new:N \l_texnegar_min_penalty_int

```

```

99 \int_new:N \l_texnegar_low_penalty_int
100 \int_new:N \l_texnegar_med_penalty_int
101 \int_new:N \l_texnegar_high_penalty_int
102 \int_new:N \l_texnegar_max_penalty_int
103
104 \int_new:N \l_fontnumber_int
105
106 \tl_new:N \l_texnegar_line_break_tl
107
108 \tl_new:N \l_texnegar_main_font_full_tl
109 \tl_new:N \l_texnegar_main_font_name_tl
110
111 \tl_new:N \l_texnegar_font_full_tl
112 \tl_new:N \l_texnegar_font_name_tl
113
114 \tl_new:N \l_texnegar_skip_default_tl
115
116 \tl_new:N \l_texnegar_active_ligs_tl
117
118 \tl_new:N \l_texnegar_gap_filler_tl
119
120 \tl_new:N \l_texnegar_use_color_tl
121 \tl_new:N \l_texnegar_color_tl
122 \tl_new:N \l_texnegar_color_rgb_tl
123
124 \dim_new:N \l_texnegar_diff_pos_dim
125
126 \bool_set_false:N \l_texnegar_kashida_fix_bool
127
128 \bool_set_false:N \l_texnegar_kashida_glyph_bool
129 \bool_set_false:N \l_texnegar_kashida_leaders_glyph_bool
130 \bool_set_false:N \l_texnegar_kashida_leaders_hrule_bool
131
132 \bool_set_false:N \l_texnegar_ligature_bool
133 \bool_set_false:N \l_texnegar_linebreakpenalty_bool
134 \bool_set_false:N \l_texnegar_hboxrecursion_bool
135 \bool_set_false:N \l_texnegar_vboxrecursion_bool
136 \bool_set_false:N \l_texnegar_color_bool
137
138 \int_set:Nn \l_texnegar_min_penalty_int { 0 }
139 \int_set:Nn \l_texnegar_low_penalty_int { 8 }
140 \int_set:Nn \l_texnegar_med_penalty_int { 15 }
141 \int_set:Nn \l_texnegar_high_penalty_int { 25 }
142 \int_set:Nn \l_texnegar_max_penalty_int { 10000 }
143
144 \tl_set:Nn \l_texnegar_stretch_glyph_tl { glyph }
145 \tl_set:Nn \l_texnegar_stretch_leaders_glyph_tl { leaders+glyph }
146 \tl_set:Nn \l_texnegar_stretch_leaders_hrule_tl { leaders+hrule }
147 \tl_set:Nn \l_texnegar_stretch_off_tl { Off }
148 \tl_set:Nn \l_texnegar_stretch_on_tl { On }
149
150 \tl_set:Nn \l_texnegar_hboxrecursion_off_tl { Off }
151 \tl_set:Nn \l_texnegar_hboxrecursion_on_tl { On }
152

```

```

153 \tl_set:Nn \l_texnegar_vboxrecursion_off_tl { Off }
154 \tl_set:Nn \l_texnegar_vboxrecursion_on_tl { On }
155
156 \tl_set:Nn \l_texnegar_fnt_kayhan_tl { kayhan }
157 \tl_set:Nn \l_texnegar_fnt_kayhannavaar_tl { kayhannavaar }
158 \tl_set:Nn \l_texnegar_fnt_kayhanpook_tl { kayhanpook }
159 \tl_set:Nn \l_texnegar_fnt_kayhansayeh_tl { kayhansayeh }
160 \tl_set:Nn \l_texnegar_fnt_khoramshahr_tl { khoramshahr }
161 \tl_set:Nn \l_texnegar_fnt_khorramshahr_tl { khorramshahr }
162 \tl_set:Nn \l_texnegar_fnt_niloofar_tl { niloofar }
163 \tl_set:Nn \l_texnegar_fnt_paatch_tl { paatch }
164 \tl_set:Nn \l_texnegar_fnt_riyaz_tl { riyaz }
165 \tl_set:Nn \l_texnegar_fnt_roya_tl { roya }
166 \tl_set:Nn \l_texnegar_fnt_shafigh_tl { shafigh }
167 \tl_set:Nn \l_texnegar_fnt_shafighKurd_tl { shafighKurd }
168 \tl_set:Nn \l_texnegar_fnt_shafighUzbek_tl { shafighUzbek }
169 \tl_set:Nn \l_texnegar_fnt_shiraz_tl { shiraz }
170 \tl_set:Nn \l_texnegar_fnt_sols_tl { sols }
171 \tl_set:Nn \l_texnegar_fnt_tabriz_tl { tabriz }
172 \tl_set:Nn \l_texnegar_fnt_titr_tl { titr }
173 \tl_set:Nn \l_texnegar_fnt_titre_tl { titre }
174 \tl_set:Nn \l_texnegar_fnt_traffic_tl { traffic }
175 \tl_set:Nn \l_texnegar_fnt_vahid_tl { vahid }
176 \tl_set:Nn \l_texnegar_fnt_vosta_tl { vosta }
177 \tl_set:Nn \l_texnegar_fnt_yaghut_tl { yaghut }
178 \tl_set:Nn \l_texnegar_fnt_yagut_tl { yagut }
179 \tl_set:Nn \l_texnegar_fnt_yas_tl { yas }
180 \tl_set:Nn \l_texnegar_fnt_yekan_tl { yekan }
181 \tl_set:Nn \l_texnegar_fnt_yermook_tl { yermook }
182 \tl_set:Nn \l_texnegar_fnt_zar_tl { zar }
183 \tl_set:Nn \l_texnegar_fnt_ziba_tl { ziba }
184 \tl_set:Nn \l_texnegar_fnt_default_tl { default }
185 \tl_set:Nn \l_texnegar_fnt_noskip_tl { noskip }
186
187 \tl_set:Nn \l_texnegar_lig_aalt_tl { aalt } % Access All Alternatives
188 \tl_set:Nn \l_texnegar_lig_ccmp_tl { ccmp } % Glyph Composition/Decomposition
189 \tl_set:Nn \l_texnegar_lig_dlig_tl { dlig } % Discretionary Ligatures
190 \tl_set:Nn \l_texnegar_lig_fina_tl { fina } % Final (Terminal) Forms
191 \tl_set:Nn \l_texnegar_lig_init_tl { init } % Initial Forms
192 \tl_set:Nn \l_texnegar_lig_locl_tl { locl } % Localized Forms
193 \tl_set:Nn \l_texnegar_lig_medi_tl { medi } % Medial Forms
194 \tl_set:Nn \l_texnegar_lig_rlig_tl { rlig } % Required Ligatures
195 \tl_set:Nn \l_texnegar_lig_default_tl { default }
196
197 \tl_set:Nn \l_texnegar_col_default_tl { magenta }
198
199 \clist_set:Nn \l_texnegar_lig_aalt_clist { } % Access All Alternatives
200 \clist_set:Nn \l_texnegar_lig_ccmp_clist { } % Glyph Composition/Decomposition
201 \clist_set:Nn \l_texnegar_lig_dlig_clist { FDF2 = , FDF3 = , FDFB = } % Discretionary
202 \clist_set:Nn \l_texnegar_lig_fina_clist { } % Final (Terminal) Forms
203 \clist_set:Nn \l_texnegar_lig_init_clist { } % Initial Forms
204 \clist_set:Nn \l_texnegar_lig_locl_clist { } % Localized Forms
205 \clist_set:Nn \l_texnegar_lig_medi_clist { } % Medial Forms
206 \clist_set:Nn \l_texnegar_lig_rlig_clist { } % Required Ligatures

```

```

207 \clist_set:Nn \l_texnegar_lig_default_clist { }
208
209 \clist_set:Nn \l_texnegar_lig_names_clist
210 {
211   \l_texnegar_lig_aalt_tl , { \l_texnegar_lig_aalt_clist } ,
212   \l_texnegar_lig_ccmp_tl , { \l_texnegar_lig_ccmp_clist } ,
213   \l_texnegar_lig_dlig_tl , { \l_texnegar_lig_dlig_clist } ,
214   \l_texnegar_lig_fina_tl , { \l_texnegar_lig_fina_clist } ,
215   \l_texnegar_lig_init_tl , { \l_texnegar_lig_init_clist } ,
216   \l_texnegar_lig_locl_tl , { \l_texnegar_lig_locl_clist } ,
217   \l_texnegar_lig_medi_tl , { \l_texnegar_lig_medi_clist } ,
218   \l_texnegar_lig_rlig_tl , { \l_texnegar_lig_rlig_clist } ,
219 }
220
221 \msg_new:nnn { texnegar } { error-kashida-character-is-not-available-in-the-main-
font }
222 {
223   Sorry,~ kashida~ character~ is~ not~ available~ in~ the~ main~ font~#1!
224 }
225
226 \msg_new:nnn { texnegar } { error-value-not-available-for-kashida-option }
227 {
228   Sorry,~ value~ ‘#1’~ is~ not~ available~ for~ ‘Kashida’~ option~ yet~!
229 }
230
231 \msg_new:nnn { texnegar } { error-specify-value-for-kashida-option }
232 {
233   Sorry,~ you~ must~ specify~ a~ value~ for~ ‘Kashida’~ option~ yet~!
234 }
235
236 \msg_new:nnn { texnegar } { warning-experimental-feature }
237 {
238   Please~ note~ that~ the~ feature~ ‘#1’~ is~ still~ experimental~
239   and~ is~ not~ regarded~ as~ stable.
240 }
241
242 \msg_new:nnn { texnegar } { hm-series-font-not-found }
243 {
244   Either~ the~ font~‘#1’~ is~ not~ installed~ on~ your~ system~ or~ does~ not~
245   belong~ to~ HM-Series-fonts.~
246   Please~ note~ that~ the~ option~ ‘Kashida=leaders+glyph’~ is~ currently~ only~
247   supported~ by~ HM-Series-fonts.~
248   If~ you~ know~ of~ any~ other~ font~ that~ supports~ this~ option,~ please~
249   let~ me~ know~ to~ add~ it~ to~ the~ list~ of~ corresponding~ fonts.~
250 }
251
252 \msg_new:nnn { texnegar } { luatex-version-is-too-old }
253 {
254   #1:~Your~luatex~is~too~old,~you~need~at~least~version~#2.#3~!
255 }
256
257 \keys_define:nn { texnegar }
258 {
259   Kashida .code:n =

```

```

260 {
261   \tl_set:Nn \l_tmpa_tl { #1 }
262   \tl_case:NnTF \l_tmpa_tl
263   {
264     \l_texnegar_stretch_glyph_tl
265     {
266       \msg_warning:nnn { texnegar } { warning-experimental-feature } { Kashida=gly
267       \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_glyph_tl }
268       \AtBeginDocument
269       {
270         \tl_set:Nx \l_texnegar_main_font_full_tl { \tex_fontname:D \tex_the:D \t
271         \tl_set:Nx \l_texnegar_main_font_name_tl { \l_texnegar_main_font_full_tl
272         \regex_replace_once:nnN { ^"([~/]+)/.* } { \1 } \l_texnegar_main_font_na
273         }
274         \bool_set_true:N \l_texnegar_kashida_fix_bool
275         \bool_set_true:N \l_texnegar_kashida_glyph_bool
276       }
277     \l_texnegar_stretch_leaders_glyph_tl
278     {
279       \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_leaders_glyph_tl
280       \bool_set_true:N \l_texnegar_kashida_fix_bool
281       \bool_set_true:N \l_texnegar_kashida_leaders_glyph_bool
282     }
283     \l_texnegar_stretch_leaders_hruler_tl
284     {
285       \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_leaders_hruler_tl
286       \bool_set_true:N \l_texnegar_kashida_fix_bool
287       \bool_set_true:N \l_texnegar_kashida_leaders_hruler_bool
288     }
289     \l_texnegar_stretch_off_tl
290     {
291       \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_off_tl }
292       \bool_set_false:N \l_texnegar_kashida_fix_bool
293     }
294     \l_texnegar_stretch_on_tl
295     {
296       \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_leaders_glyph_tl
297       \bool_set_true:N \l_texnegar_kashida_fix_bool
298       \bool_set_true:N \l_texnegar_kashida_leaders_glyph_bool
299     }
300   } { } { \tl_set:Nx \l_texnegar_gap_filler_tl { #1 } }
301   \tl_if_empty:NT \l_texnegar_gap_filler_tl { \msg_error:nn { texnegar } { error-
specify-value-for-kashida-option } }
302 } ,
303
304 linebreakpenalty .code:n =
305 {
306   \int_set:Nn \l_tmpa_int { #1 }
307   \int_case:nnTF \l_tmpa_int
308   {
309     \l_texnegar_min_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
310     \l_texnegar_low_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
311     \l_texnegar_med_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
312     \l_texnegar_high_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {

```



```

313         \l_texnegar_max_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
314     } } { \int_set:Nn \l_texnegar_line_break_penalty_int { #1 } }
315     \bool_set_true:N \l_texnegar_linebreakpenalty_bool
316 } ,
317
318 kashidastretch .code:n =
319 {
320     \tl_set:Nn \l_tmpa_tl { #1 }
321     \tl_case:NnTF \l_tmpa_tl
322     {
323         \l_texnegar_fnt_kayhan_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
324         \l_texnegar_fnt_kayhannavaar_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
325         \l_texnegar_fnt_kayhanpook_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
326         \l_texnegar_fnt_kayhansayeh_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
327         \l_texnegar_fnt_khoramshahr_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
328         \l_texnegar_fnt_khorramshahr_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
329         \l_texnegar_fnt_nilooфар_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
330         \l_texnegar_fnt_paatch_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
331         \l_texnegar_fnt_riyaz_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
332         \l_texnegar_fnt_roya_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
333         \l_texnegar_fnt_shafigh_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
334         \l_texnegar_fnt_shafighKurd_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
335         \l_texnegar_fnt_shafighUzbek_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
336         \l_texnegar_fnt_shiraz_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
337         \l_texnegar_fnt_sols_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
338         \l_texnegar_fnt_tabriz_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.11
339         \l_texnegar_fnt_titr_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
340         \l_texnegar_fnt_titre_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
341         \l_texnegar_fnt_traffic_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
342         \l_texnegar_fnt_vahid_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
343         \l_texnegar_fnt_vosta_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
344         \l_texnegar_fnt_yaghut_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
345         \l_texnegar_fnt_yagut_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
346         \l_texnegar_fnt_yas_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
347         \l_texnegar_fnt_yekan_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
348         \l_texnegar_fnt_yermook_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
349         \l_texnegar_fnt_zar_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.11
350         \l_texnegar_fnt_ziba_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.11
351         \l_texnegar_fnt_default_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
352         \l_texnegar_fnt_noskip_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0
353     } } { \tl_set:Nn \l_texnegar_skip_default_tl { #1 } }
354 } ,
355 kashidastretch .default:n = \tl_set:Nn \l_texnegar_skip_default_tl { 0 em plus 0.5 em }
356
357 ligatures .code:n =
358 {
359     \tl_set:Nn \l_tmpa_tl { #1 }
360     \tl_case:NnTF \l_tmpa_tl
361     {
362         \l_texnegar_lig_aalt_tl { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
363         \l_texnegar_lig_ccmp_tl { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
364         \l_texnegar_lig_dlig_tl { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
365         \l_texnegar_lig_fina_tl { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
366         \l_texnegar_lig_init_tl { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar

```

```

367         \l_texnegar_lig_locl_tl { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
368         \l_texnegar_lig_medi_tl { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
369         \l_texnegar_lig_rlig_tl { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
370         \l_texnegar_lig_default_tl { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
371     } { } { \tl_set:Nn \l_texnegar_active_ligs_tl { #1 } }
372     \bool_set_true:N \l_texnegar_ligature_bool
373 } ,
374 ligatures .default:n = \tl_set:Nn \l_texnegar_active_ligs_tl { \l_texnegar_lig_default_t
375
376 color .code:n =
377 {
378     \tl_set:Nn \l_tmpa_tl { #1 }
379     \tl_if_empty:NTF \l_tmpa_tl
380     {
381         \tl_set:Nx \l_texnegar_color_tl { \l_texnegar_col_default_tl }
382     }
383     {
384         \tl_set:Nx \l_texnegar_color_tl { \l_tmpa_tl }
385     }
386     \bool_set_true:N \l_texnegar_color_bool
387     \sys_if_engine luatex:T
388     {
389         \convertcolourspec{named}{\l_texnegar_color_tl}{rgb}\l_texnegar_color_rgb_tl
390         \directlua{\l_texnegar_color_rgb_tl = "\l_texnegar_color_rgb_tl"}
391     }
392 } ,
393
394 hboxrecursion .code:n =
395 {
396     \tl_set:Nn \l_tmpa_tl { #1 }
397     \tl_case:NnTF \l_tmpa_tl
398     {
399         \l_texnegar_hboxrecursion_off_tl
400         {
401             \bool_set_false:N \l_texnegar_hboxrecursion_bool
402         }
403         \l_texnegar_hboxrecursion_on_tl
404         {
405             \bool_set_true:N \l_texnegar_hboxrecursion_bool
406         }
407     } { } { \bool_set_false:N \l_texnegar_hboxrecursion_bool }
408 } ,
409 hboxrecursion .default:n = \bool_set_true:N \l_texnegar_hboxrecursion_bool ,
410
411 vboxrecursion .code:n =
412 {
413     \tl_set:Nn \l_tmpa_tl { #1 }
414     \tl_case:NnTF \l_tmpa_tl
415     {
416         \l_texnegar_vboxrecursion_off_tl
417         {
418             \bool_set_false:N \l_texnegar_vboxrecursion_bool
419         }
420         \l_texnegar_vboxrecursion_on_tl

```

```

421         {
422             \bool_set_true:N \l_texnegar_vboxrecursion_bool
423         }
424     } { } { \bool_set_false:N \l_texnegar_vboxrecursion_bool }
425 },
426 vboxrecursion .default:n = \bool_set_true:N \l_texnegar_vboxrecursion_bool ,
427 }
428
429 \ProcessKeysOptions { texnegar }
430
431 \sys_if_engine luatex:T
432 {
433     \NewDocumentCommand \KashidaHMFixOff {} { \directlua{StopStretching()} }
434     \NewDocumentCommand \KashidaHMFixOn {} { \directlua{StartStretching()} }
435 }
436
437 \sys_if_engine xetex:T
438 {
439     \NewDocumentCommand \KashidaHMFixOn {} { \bool_set_true:N \l_texnegar_kashida_fix_bool }
440     \NewDocumentCommand \KashidaHMFixOff {} { \bool_set_false:N \l_texnegar_kashida_fix_bool }
441 }
442
443 \tex_let:D \KashidaOn \KashidaHMFixOn
444 \tex_let:D \KashidaOff \KashidaHMFixOff
445
446 \bool_if:NTF \l_texnegar_kashida_fix_bool
447 {
448     \tl_if_empty:NT \l_texnegar_skip_default_tl { \tl_set:Nn \l_texnegar_skip_default_tl {
449     }
450     {
451         \tl_set:NV \l_texnegar_skip_default_tl \c_texnegar_skip_a_tl
452     }
453 }
454 %% % \makeatletter
455 %% % \newif\if@Kashida@on
456 %% % Becuase Vafa Khalighi has copied the above code (injecting the character uni+200E) in xepersian.
457 %% % (https://tug.org/svn/texlive/trunk/Master/texmf-dist/tex/xelatex/xepersian/kashida-
458 %% % xepersian.def?revision=55165&view=co),
459 %% % the following line of code is not needed in xepersian anymore.
460 %% % \newif\if@Kashida@XB@fix
461 %% % \makeatother
462 \endinput
463 </texnegar-ini-tex>

```

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

```

464 <*texnegar-common-kashida-tex>
465 \ProvidesExplFile {texnegar-common-kashida.tex} {2020-08-30} {0.1b} { Full implementation of
466
467 \bool_if:NT \l_texnegar_ligature_bool
468 {
469     \clist_new:N \l_texnegar_ligatures_clist
470     \int_new:N \l_texnegar_lig_names_len_int

```

```

\int_set:Nn \l_texnegar_lig_names_len_int { \clist_count:N \l_texnegar_lig_names_clist }
\int_step_inline:nnnn { 1 } { 2 } { \l_texnegar_lig_names_len_int }
{
  \int_set:Nn \l_tmpa_int { #1 }
  \int_set:Nn \l_tmppb_int { \int_eval:n { \l_tmpa_int + 1 } }
  \tl_set:Nf \l_tmpa_tl { \clist_item:Nn \l_texnegar_lig_names_clist { \l_tmpa_int } }
  \clist_set:Nx \l_tmpa_clist { { \clist_item:Nn \l_texnegar_lig_names_clist { \l_tmppb_i
  \bool_if:nT { \tl_if_eq_p:NN \l_texnegar_active_ligs_tl \l_tmpa_tl || \tl_if_eq_p:NN
    {
      \clist_put_left:Nx \l_texnegar_ligatures_clist { \l_tmpa_clist }
    }
  }
}
\clist_map_inline:Nn \l_texnegar_ligatures_clist
{
  \seq_set_split:Nnn \l_tmpa_seq { = } { #1 }
  \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl { } { }
  \seq_pop_left:NN \l_tmpa_seq \l_tmppb_tl { } { }
  \tl_const:cx { \tl_use:N \l_tmppb_tl } { \char"\l_tmpa_tl \ }
}
}
}
\bool_if:NT \l_texnegar_linebreakpenalty_bool
{
  %% Partly adapted from LaTeX2e source
  \cs_new:Nn \texnegar_line_break: {
    \if_mode_vertical:
      \GenericError{
        \space\space\space\space\space\space\space\space\space\space\space\space\space\space\space\space
      ){
        LaTeX Error: Theres no line here to end
      ){
        See the LaTeX manual or LaTeX Companion for explanation.
      ){
        Your command was ignored.\MessageBreak
        Type \space I <command> <return> \space to replace it~
        with another command,\MessageBreak
        or \space <return> \space to continue without it.}
      }
    \else:
      \l_tmpa_skip \tex_lastskip:D
      \tex_unskip:D
      \tex_penalty:D -\l_texnegar_line_break_penalty_int
      \dim_compare:nT { \l_tmpa_skip > \c_zero_skip }
      { \skip_horizontal:N \l_tmpa_skip \tex_ignorespaces:D }
    \fi:
  }
}
\NewDocumentCommand { \discouragebadlinebreaks } { 0{\l_texnegar_line_break_penalty_int} 0
{
  \IfNoValueF {#1}{
    { \int_set:Nn \l_texnegar_line_break_penalty_int {#1} }
  }
  \IfNoValueF {#2}{
    { \tl_set:Nn \l_texnegar_skip_default_tl {#2} }
  }
  \texnegar_put_line_breaks:n { #3 }
}

```

```

525
526 \cs_new_protected:Nn \texnegar_put_line_breaks:n
527 {
528   \tl_set:Nn \l_texnegar_line_break_tl { #1 }
529   \regex_replace_all:nnN { ([\])+ } { \ \0 \ \c{texnegar_line_break:}\ } \l_texnegar_line_break_tl
530   \tl_use:N \l_texnegar_line_break_tl
531 }
532 }
533 </texnegar-common-kashida-tex>

```

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

```

534 <*texnegar-xetex-kashida-tex>
535 \ProvidesExplFile {texnegar-xetex-kashida.tex} {2020-08-30} {0.1b} { Full implementation of
536
537 \newXeTeXintercharclass \c_texnegar_d_charclass % dual-joiner class
538 \newXeTeXintercharclass \c_texnegar_l_charclass % lam
539 \newXeTeXintercharclass \c_texnegar_r_charclass % right-joiner
540 \newXeTeXintercharclass \c_texnegar_a_charclass % alef
541 \newXeTeXintercharclass \c_texnegar_y_charclass % yeh
542
543 \tex_input:D { texnegar-common-kashida.tex }
544
545 \tl_set:Nn \l_texnegar_use_color_tl
546 {
547   \bool_if:NTF \l_texnegar_color_bool
548   {
549     \colorlet{default}{\l_texnegar_color_tl}
550   }
551   {
552     \colorlet{default}{.}
553   }
554   \color{default}
555 }
556
557 %% Partly adapted from the code provided by David Carlisle in:
558 %% https://tex.stackexchange.com/questions/356709/how-to-know-the-width-and-fill-the-glue-space-between-two-characters-when-using/356721#356721
559 \cs_new:Npn \texnegar_kashida_glyph #1
560 {
561   \bool_if:NT \l_texnegar_kashida_fix_bool
562   {
563     \c_texnegar_lrm_int\tex_penalty:D 10000
564     \mode_leave_vertical:
565     \tex_global:D \tex_advance:D \l_texnegar_counter_int \c_one_int
566
567     \tl_set:Nx \l_texnegar_pos_tl { p\tex_romannumeral:D \l_texnegar_counter_int }
568     \tl_set:Nx \l_texnegar_zref_tl { z\tex_romannumeral:D \l_texnegar_counter_int }
569
570     \zsaveposx{x_i\l_texnegar_zref_tl}
571     \tl_set:Nx \l_tmpa_tl
572     {
573       \iow_now:cx { @auxout }
574     }
575     \token_to_str:N \gdef \exp_after:wN \token_to_str:N \cs:w xi\l_texnegar_pos_tl \cs

```

```

576     }
577   }
578   \l_tmpa_tl
579   \skip_horizontal:n { #1 }
580   \zsaveposx{x_f\l_texpnegar_zref_tl}
581   \tl_set:Nx \l_tmpa_tl
582   {
583     \iow_now:cx { @auxout }
584     {
585       \token_to_str:N \gdef \exp_after:wN \token_to_str:N \cs:w xf\l_texpnegar_pos_tl \cs
586     }
587   }
588   \l_tmpa_tl
589   \exp_after:wN
590   \if_meaning:w
591     \cs:w xi\l_texpnegar_pos_tl \cs_end: \tex_relax:D
592   \else:
593     \dim_set:Nn \l_texpnegar_diff_pos_dim
594     {
595       \dim_eval:n { \cs:w xi\l_texpnegar_pos_tl \cs_end: sp - \cs:w xf\l_texpnegar_pos_tl
596     }
597     \dim_compare:nTF { \l_texpnegar_diff_pos_dim == 0sp }
598     { }
599     { \llap { \resizebox { \l_texpnegar_diff_pos_dim \tex_relax:D } { \height } { \l_texp
600   \fi:
601 }
602 }
603
604 \cs_new:Npn \texpnegar_kashida_leaders #1
605 {
606   \bool_if:NT \l_texpnegar_kashida_fix_bool
607   {
608     \tl_if_eq:NNTF \l_texpnegar_gap_filler_tl \l_texpnegar_stretch_leaders_glyph_tl
609     {
610       \tl_set:Nx \l_texpnegar_font_full_tl { \tex_fontname:D \tex_the:D \tex_font:D }
611       \tl_set:Nx \l_texpnegar_font_name_tl { \l_texpnegar_font_full_tl }
612       \tl_set:Nx \l_texpnegar_font_init_tl { \l_texpnegar_font_name_tl }
613       \regex_replace_once:nnN { ^" \[?(HM)[\ \ ](X|F).* } { \1\2 } \l_texpnegar_font_init
614       \tl_set:Nn \l_tmpa_tl { HMF }
615       \tl_set:Nn \l_tmpb_tl { HMX }
616       \bool_if:nTF { \str_if_eq_p:NN { \l_texpnegar_font_init_tl } { \l_tmpa_tl } || \str
617       {
618         \hbox_set:Nn \l_texpnegar_ksh_box { \l_texpnegar_use_color_tl \XeTeXglyph\XeTeXg
619         \c_texpnegar_zwj_int \tex_penalty:D 10000
620         \tex_leaders:D \copy\l_texpnegar_ksh_box \skip_horizontal:n { #1 }
621         \c_texpnegar_zwj_int
622       }
623       {
624         \msg_error:nxx { texpnegar } { hm-series-font-not-found } { \l_texpnegar_font_na
625       }
626     }
627   }
628   %% Partly adapted from the code provided by Jonathan Kew in:
629   %% https://tug.org/pipermail/xetex/2009-February/012307.html.

```

```

630         %% Somebody notified me that the code in 'kashida-xepersian.def' from xepersian
631         %% package is an exact copy of Jonathan Kew's code. Being unaware of this, in
632         %% the earlier versions of this package I made a mistake and acknowledged
633         %% Vafa Khalighi instead of Jonathan Kew. A sincere thank you to Jonathan Kew
634         %% for his excellent code.
635         \c_texnegar_lrm_int\c_texnegar_zwj_int
636         {\l_texnegar_use_color_tl\tex_penalty:D 10000
637         \tex_leaders:D \tex_hrulerule:D height \XeTeXglyphbounds \c_texnegar_two_int
638         \int_use:N \XeTeXcharglyph \c_texnegar_ksh_int depth \XeTeXglyphbounds \c_texnegar
639         \int_use:N \XeTeXcharglyph \c_texnegar_ksh_int \skip_horizontal:n { #1 }
640         }
641         \c_texnegar_zwj_int
642     }
643 }
644 }
645
646 \XeTeXinterchartokenstate = 1
647
648 \clist_set:Nn \l_texnegar_a_clist { 0622,0623,0625,0627 } %
649 \clist_map_inline:Nn \l_texnegar_a_clist
650 {
651     \XeTeXcharclass "#1 \c_texnegar_a_charclass
652 }
653
654 \clist_set:Nn \l_texnegar_d_clist { 0626,0628,062A,062B,062C,062D,062E,0633,0634,0635,0636,0
655 \clist_map_inline:Nn \l_texnegar_d_clist
656 {
657     \XeTeXcharclass "#1 \c_texnegar_d_charclass
658 }
659
660 \clist_set:Nn \l_texnegar_l_clist { 0644 } %
661 \clist_map_inline:Nn \l_texnegar_l_clist
662 {
663     \XeTeXcharclass "#1 \c_texnegar_l_charclass
664 }
665
666 \clist_set:Nn \l_texnegar_r_clist { 0624,0629,062F,0630,0631,0632,0648,0698 } % , , , , ,
667 \clist_map_inline:Nn \l_texnegar_r_clist
668 {
669     \XeTeXcharclass "#1 \c_texnegar_r_charclass
670 }
671
672 \clist_set:Nn \l_texnegar_y_clist { 0649,064A,06CC } % , ,
673 \clist_map_inline:Nn \l_texnegar_y_clist
674 {
675     \XeTeXcharclass "#1 \c_texnegar_y_charclass
676 }
677
678 \tl_if_eq:NNTF \l_texnegar_gap_filler_tl \l_texnegar_stretch_glyph_tl {
679     \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_y_charclass = {
680         \bool_if:NNTF \l_texnegar_kashida_fix_bool
681         { \c_texnegar_zwj_int \texnegar_kashida_glyph \l_texnegar_skip_default_tl \c_texnegar_zw
682         { \c_texnegar_zwj_int \texnegar_kashida_glyph \c_texnegar_skip_a_tl \c_texnegar_zwj_int
683     }

```

```

684 \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_y_charclass = {
685   \bool_if:NTF \l_texnegar_kashida_fix_bool
686   { \c_texnegar_zwj_int \texnegar_kashida_glyph \l_texnegar_skip_default_tl \c_texnegar_zwj_int
687     { \c_texnegar_zwj_int \texnegar_kashida_glyph \c_texnegar_skip_a_tl \c_texnegar_zwj_int
688   }
689 \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_d_charclass = { \c_texnegar_zwj_int
690 \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_d_charclass = { \c_texnegar_zwj_int
691 \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_d_charclass = { \c_texnegar_zwj_int
692 \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_l_charclass = { \c_texnegar_zwj_int
693 \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_l_charclass = { \c_texnegar_zwj_int
694 \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_r_charclass = { \c_texnegar_zwj_int
695 \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_a_charclass = { \c_texnegar_zwj_int
696 \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_r_charclass = { \c_texnegar_zwj_int
697 \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_a_charclass = { }
698 }
699 {
700   \bool_if:NTF {
701     \tl_if_eq_p:NN \l_texnegar_gap_filler_tl \l_texnegar_stretch_leaders_glyph_tl ||
702     \tl_if_eq_p:NN \l_texnegar_gap_filler_tl \l_texnegar_stretch_leaders_hruler_tl
703   }
704   {
705     \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_y_charclass = {
706       \bool_if:NTF \l_texnegar_kashida_fix_bool
707       { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
708       { \texnegar_kashida_leaders \c_texnegar_skip_a_tl }
709     }
710     \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_y_charclass = {
711       \bool_if:NTF \l_texnegar_kashida_fix_bool
712       { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
713       { \texnegar_kashida_leaders \c_texnegar_skip_a_tl }
714     }
715     \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_d_charclass = { \texnegar_kashida_leaders
716     \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_d_charclass = { \texnegar_kashida_leaders
717     \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_d_charclass = { \texnegar_kashida_leaders
718     \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_l_charclass = { \texnegar_kashida_leaders
719     \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_l_charclass = { \texnegar_kashida_leaders
720     \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_r_charclass = { \texnegar_kashida_leaders
721     \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_a_charclass = { \texnegar_kashida_leaders
722     \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_r_charclass = { \texnegar_kashida_leaders
723     \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_a_charclass = { }
724   }
725   {
726     \msg_error:nxx { texnegar } { error-value-not-available-for-kashida-option } { \l_texnegar_kashida_fix_bool }
727   }
728 }
729
730 \endinput
731 </texnegar-xetex-kashida-tex>

```

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

```

732 < *texnegar-char-table.lua >
733 --
734 -- This is file 'texnegar-char-table.lua',
735 -- generated with the docstrip utility.

```



```

736 --
737 -- The original source files were:
738 --
739 -- texnegar.dtx (with options: 'texnegar-char-table-lua')
740 --
741 -- Copyright (C) 2020 Hossein Movahhedian
742 --
743 -- It may be distributed and/or modified under the LaTeX Project Public License,
744 -- version 1.3c or higher (your choice). The latest version of
745 -- this license is at: http://www.latex-project.org/lppl.txt
746 --
747 -- texnegar_char_table = texnegar_char_table or {}
748 -- local texnegar_char_table = texnegar_char_table
749 -- texnegar_char_table.module = {
750 --     name = "texnegar_char_table",
751 --     version = "0.1b",
752 --     date = "2020-08-30",
753 --     description = "Full implementation of kashida feature in XeLaTeX and LuaLaTeX",
754 --     author = "Hossein Movahhedian",
755 --     copyright = "Hossein Movahhedian",
756 --     license = "LPPL v1.3c"
757 -- }
758 --
759 -- -- ^A%% texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
760 -- local err, warn, info, log = luatexbase.provides_module(texnegar_char_table.module)
761 -- texnegar_char_table.log = log or (function (s) luatexbase.module_info("texnegar_char_table", s))
762 -- texnegar_char_table.warning = warn or (function (s) luatexbase.module_warning("texnegar_char_table", s))
763 -- texnegar_char_table.error = err or (function (s) luatexbase.module_error("texnegar_char_table", s))
764 --
765 local peCharTableInitial = {
766     [64344] = utf8.char(64344), -- "", utf8.codepoint("") == 64344, "\u{FB58}", INITIAL FOR
767     [64380] = utf8.char(64380), -- "", utf8.codepoint("") == 64380, "\u{FB7C}", INITIAL FOR
768     [64400] = utf8.char(64400), -- "", utf8.codepoint("") == 64400, "\u{FB90}", INITIAL FOR
769     [64404] = utf8.char(64404), -- "", utf8.codepoint("") == 64404, "\u{FB94}", INITIAL FOR
770     [64510] = utf8.char(64510), -- "", utf8.codepoint("") == 64510, "\u{FBFE}", INITIAL FOR
771     [65169] = utf8.char(65169), -- "", utf8.codepoint("") == 65169, "\u{FE91}", INITIAL FOR
772     [65175] = utf8.char(65175), -- "", utf8.codepoint("") == 65175, "\u{FE97}", INITIAL FOR
773     [65179] = utf8.char(65179), -- "", utf8.codepoint("") == 65179, "\u{FE9B}", INITIAL FOR
774     [65183] = utf8.char(65183), -- "", utf8.codepoint("") == 65183, "\u{FE9F}", INITIAL FOR
775     [65187] = utf8.char(65187), -- "", utf8.codepoint("") == 65187, "\u{FEA3}", INITIAL FOR
776     [65191] = utf8.char(65191), -- "", utf8.codepoint("") == 65191, "\u{FEA7}", INITIAL FOR
777     [65203] = utf8.char(65203), -- "", utf8.codepoint("") == 65203, "\u{FEB3}", INITIAL FOR
778     [65207] = utf8.char(65207), -- "", utf8.codepoint("") == 65207, "\u{FEB7}", INITIAL FOR
779     [65211] = utf8.char(65211), -- "", utf8.codepoint("") == 65211, "\u{FEBB}", INITIAL FOR
780     [65215] = utf8.char(65215), -- "", utf8.codepoint("") == 65215, "\u{FEBF}", INITIAL FOR
781     [65219] = utf8.char(65219), -- "", utf8.codepoint("") == 65219, "\u{FEC3}", INITIAL FOR
782     [65223] = utf8.char(65223), -- "", utf8.codepoint("") == 65223, "\u{FEC7}", INITIAL FOR
783     [65227] = utf8.char(65227), -- "", utf8.codepoint("") == 65227, "\u{FECB}", INITIAL FOR
784     [65231] = utf8.char(65231), -- "", utf8.codepoint("") == 65231, "\u{FECF}", INITIAL FOR
785     [65235] = utf8.char(65235), -- "", utf8.codepoint("") == 65235, "\u{FED3}", INITIAL FOR
786     [65239] = utf8.char(65239), -- "", utf8.codepoint("") == 65239, "\u{FED7}", INITIAL FOR
787     [65243] = utf8.char(65243), -- "", utf8.codepoint("") == 65243, "\u{FEDB}", INITIAL FOR
788     [65247] = utf8.char(65247), -- "", utf8.codepoint("") == 65247, "\u{FEDF}", INITIAL FOR
789     [65251] = utf8.char(65251), -- "", utf8.codepoint("") == 65251, "\u{FEE3}", INITIAL FOR

```

```

790 [65255] = utf8.char(65255), -- "", utf8.codepoint("") == 65255, "\u{FEE7}", INITIAL FORM
791 [65259] = utf8.char(65259), -- "", utf8.codepoint("") == 65259, "\u{FEEB}", INITIAL FORM
792 [65267] = utf8.char(65267), -- "", utf8.codepoint("") == 65267, "\u{FEF3}", INITIAL FORM
793 }
794
795 local peCharTableMedial = {
796 [1600] = utf8.char(1600), -- "", utf8.codepoint("") == 1600, "\u{0640}", ARABIC TATW
797 [64345] = utf8.char(64345), -- "", utf8.codepoint("") == 64345, "\u{FB59}", MEDIAL FORM
798 [64381] = utf8.char(64381), -- "", utf8.codepoint("") == 64381, "\u{FB7D}", MEDIAL FORM
799 [64401] = utf8.char(64401), -- "", utf8.codepoint("") == 64401, "\u{FB91}", MEDIAL FORM
800 [64405] = utf8.char(64405), -- "", utf8.codepoint("") == 64405, "\u{FB95}", MEDIAL FORM
801 [64425] = utf8.char(64425), -- "", utf8.codepoint("") == 64425, "\u{FBA9}", MEDIAL FORM
802 [64429] = utf8.char(64429), -- "", utf8.codepoint("") == 64429, "\u{FBAD}", MEDIAL FORM
803 [64511] = utf8.char(64511), -- "", utf8.codepoint("") == 64511, "\u{FBFF}", MEDIAL FORM
804 [65170] = utf8.char(65170), -- "", utf8.codepoint("") == 65170, "\u{FE92}", MEDIAL FORM
805 [65176] = utf8.char(65176), -- "", utf8.codepoint("") == 65176, "\u{FE98}", MEDIAL FORM
806 [65180] = utf8.char(65180), -- "", utf8.codepoint("") == 65180, "\u{FE9C}", MEDIAL FORM
807 [65184] = utf8.char(65184), -- "", utf8.codepoint("") == 65184, "\u{FEA0}", MEDIAL FORM
808 [65188] = utf8.char(65188), -- "", utf8.codepoint("") == 65188, "\u{FEA4}", MEDIAL FORM
809 [65192] = utf8.char(65192), -- "", utf8.codepoint("") == 65192, "\u{FEA8}", MEDIAL FORM
810 [65204] = utf8.char(65204), -- "", utf8.codepoint("") == 65204, "\u{FEB4}", MEDIAL FORM
811 [65208] = utf8.char(65208), -- "", utf8.codepoint("") == 65208, "\u{FEB8}", MEDIAL FORM
812 [65212] = utf8.char(65212), -- "", utf8.codepoint("") == 65212, "\u{FEB C}", MEDIAL FORM
813 [65216] = utf8.char(65216), -- "", utf8.codepoint("") == 65216, "\u{FEC0}", MEDIAL FORM
814 [65220] = utf8.char(65220), -- "", utf8.codepoint("") == 65220, "\u{FEC4}", MEDIAL FORM
815 [65224] = utf8.char(65224), -- "", utf8.codepoint("") == 65224, "\u{FEC8}", MEDIAL FORM
816 [65228] = utf8.char(65228), -- "", utf8.codepoint("") == 65228, "\u{FECC}", MEDIAL FORM
817 [65232] = utf8.char(65232), -- "", utf8.codepoint("") == 65232, "\u{FED0}", MEDIAL FORM
818 [65236] = utf8.char(65236), -- "", utf8.codepoint("") == 65236, "\u{FED4}", MEDIAL FORM
819 [65240] = utf8.char(65240), -- "", utf8.codepoint("") == 65240, "\u{FED8}", MEDIAL FORM
820 [65244] = utf8.char(65244), -- "", utf8.codepoint("") == 65244, "\u{FEDC}", MEDIAL FORM
821 [65248] = utf8.char(65248), -- "", utf8.codepoint("") == 65248, "\u{FEE0}", MEDIAL FORM
822 [65252] = utf8.char(65252), -- "", utf8.codepoint("") == 65252, "\u{FEE4}", MEDIAL FORM
823 [65256] = utf8.char(65256), -- "", utf8.codepoint("") == 65256, "\u{FEE8}", MEDIAL FORM
824 [65260] = utf8.char(65260), -- "", utf8.codepoint("") == 65260, "\u{FEEC}", MEDIAL FORM
825 [65268] = utf8.char(65268), -- "", utf8.codepoint("") == 65268, "\u{FEF4}", MEDIAL FORM
826 }
827
828 local peCharTableFinal = {
829 [64343] = utf8.char(64343), -- "", utf8.codepoint("") == 64343, "\u{FB57}", FINAL FORM
830 [64379] = utf8.char(64379), -- "", utf8.codepoint("") == 64379, "\u{FB7B}", FINAL FORM
831 [64395] = utf8.char(64395), -- "", utf8.codepoint("") == 64395, "\u{FB8B}", FINAL FORM
832 [64399] = utf8.char(64399), -- "", utf8.codepoint("") == 64399, "\u{FB8F}", FINAL FORM
833 [64403] = utf8.char(64403), -- "", utf8.codepoint("") == 64403, "\u{FB93}", FINAL FORM
834 [64421] = utf8.char(64421), -- "", utf8.codepoint("") == 64421, "\u{FBA5}", FINAL FORM
835 [64509] = utf8.char(64509), -- "", utf8.codepoint("") == 64509, "\u{FBFD}", FINAL FORM
836 [65166] = utf8.char(65166), -- "", utf8.codepoint("") == 65166, "\u{FE8E}", FINAL FORM
837 [65168] = utf8.char(65168), -- "", utf8.codepoint("") == 65168, "\u{FE90}", FINAL FORM
838 [65172] = utf8.char(65172), -- "", utf8.codepoint("") == 65172, "\u{FE94}", FINAL FORM
839 [65174] = utf8.char(65174), -- "", utf8.codepoint("") == 65174, "\u{FE96}", FINAL FORM
840 [65178] = utf8.char(65178), -- "", utf8.codepoint("") == 65178, "\u{FE9A}", FINAL FORM
841 [65182] = utf8.char(65182), -- "", utf8.codepoint("") == 65182, "\u{FE9E}", FINAL FORM
842 [65186] = utf8.char(65186), -- "", utf8.codepoint("") == 65186, "\u{FEA2}", FINAL FORM
843 [65190] = utf8.char(65190), -- "", utf8.codepoint("") == 65190, "\u{FEA6}", FINAL FORM

```

```

844 [65194] = utf8.char(65194), -- "", utf8.codepoint("") == 65194, "\u{FEAA}", FINAL FORM
845 [65196] = utf8.char(65196), -- "", utf8.codepoint("") == 65196, "\u{FEAC}", FINAL FORM
846 [65198] = utf8.char(65198), -- "", utf8.codepoint("") == 65198, "\u{FEAE}", FINAL FORM
847 [65200] = utf8.char(65200), -- "", utf8.codepoint("") == 65200, "\u{FEB0}", FINAL FORM
848 [65202] = utf8.char(65202), -- "", utf8.codepoint("") == 65202, "\u{FEB2}", FINAL FORM
849 [65206] = utf8.char(65206), -- "", utf8.codepoint("") == 65206, "\u{FEB6}", FINAL FORM
850 [65210] = utf8.char(65210), -- "", utf8.codepoint("") == 65210, "\u{FEBA}", FINAL FORM
851 [65214] = utf8.char(65214), -- "", utf8.codepoint("") == 65214, "\u{FEBE}", FINAL FORM
852 [65218] = utf8.char(65218), -- "", utf8.codepoint("") == 65218, "\u{FEC2}", FINAL FORM
853 [65222] = utf8.char(65222), -- "", utf8.codepoint("") == 65222, "\u{FEC6}", FINAL FORM
854 [65226] = utf8.char(65226), -- "", utf8.codepoint("") == 65226, "\u{FECA}", FINAL FORM
855 [65230] = utf8.char(65230), -- "", utf8.codepoint("") == 65230, "\u{FECE}", FINAL FORM
856 [65234] = utf8.char(65234), -- "", utf8.codepoint("") == 65234, "\u{FED2}", FINAL FORM
857 [65238] = utf8.char(65238), -- "", utf8.codepoint("") == 65238, "\u{FED6}", FINAL FORM
858 [65242] = utf8.char(65242), -- "", utf8.codepoint("") == 65242, "\u{FEDA}", FINAL FORM
859 [65246] = utf8.char(65246), -- "", utf8.codepoint("") == 65246, "\u{FEDE}", FINAL FORM
860 [65250] = utf8.char(65250), -- "", utf8.codepoint("") == 65250, "\u{FEE2}", FINAL FORM
861 [65254] = utf8.char(65254), -- "", utf8.codepoint("") == 65254, "\u{FEE6}", FINAL FORM
862 [65258] = utf8.char(65258), -- "", utf8.codepoint("") == 65258, "\u{FEEA}", FINAL FORM
863 [65262] = utf8.char(65262), -- "", utf8.codepoint("") == 65262, "\u{FEEE}", FINAL FORM
864 [65264] = utf8.char(65264), -- "", utf8.codepoint("") == 65264, "\u{FEF0}", FINAL FORM
865 [65266] = utf8.char(65266), -- "", utf8.codepoint("") == 65266, "\u{FEF2}", FINAL FORM
866 [65276] = utf8.char(65276), -- "", utf8.codepoint("") == 65276, "\u{FEFC}", FINAL FORM
867 }
868
869 return peCharTableInitial, peCharTableMedial, peCharTableFinal
870 --
871 --
872 -- End of file 'texnegar-char-table.lua'.
873 \>/texnegar-char-table.lua

```

## 1.8 File: texnegar.lua

```

874 \>*texnegar.lua
875 --
876 -- This is file 'texnegar.lua',
877 -- generated with the docstrip utility.
878 --
879 -- The original source files were:
880 --
881 -- texnegar.dtx (with options: 'texnegar-lua')
882 --
883 -- Copyright (C) 2020 Hossein Movahhedian
884 --
885 -- It may be distributed and/or modified under the LaTeX Project Public License,
886 -- version 1.3c or higher (your choice). The latest version of
887 -- this license is at: http://www.latex-project.org/lppl.txt
888 --
889 -- texnegar = texnegar or {}
890 -- local texnegar = texnegar
891 -- texnegar.module = {
892 --     name = "texnegar",
893 --     version = "0.1b",
894 --     date = "2020-08-30",
895 --     description = "Full implementation of kashida feature in XeLaTeX and LuaLaTeX",

```

```

896 --      author      = "Hossein Movahhedian",
897 --      copyright   = "Hossein Movahhedian",
898 --      license      = "LPPL v1.3c"
899 -- }
900 --
901 -- -- ^A%% texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
902 -- local err, warn, info, log = luatexbase.provides_module(texnegar.module)
903 -- texnegar.log      = log or (function (s) luatexbase.module_info("texnegar", s) end)
904 -- texnegar.warning = warn or (function (s) luatexbase.module_warning("texnegar", s) end)
905 -- texnegar.error   = err or (function (s) luatexbase.module_error("texnegar", s) end)
906
907 tex.enableprimitives('',tex.extraprimitives ())
908 dofile(kpse.find_file("texnegar-ini.lua"))
909 --
910 --
911 -- End of file 'texnegar.lua'.
912 </texnegar-lua>

```

## 1.9 File: texnegar-ini.lua

```

913 <*texnegar-ini-lua>
914 --
915 -- This is file 'texnegar-ini.lua',
916 -- generated with the docstrip utility.
917 --
918 -- The original source files were:
919 --
920 -- texnegar.dtx (with options: 'texnegar-ini-lua')
921 --
922 -- Copyright (C) 2020 Hossein Movahhedian
923 --
924 -- It may be distributed and/or modified under the LaTeX Project Public License,
925 -- version 1.3c or higher (your choice). The latest version of
926 -- this license is at: http://www.latex-project.org/lppl.txt
927 --
928 -- texnegar_ini      = texnegar_ini or {}
929 -- local texnegar_ini = texnegar_ini
930 -- texnegar_ini.module = {
931 --     name          = "texnegar_ini",
932 --     version       = "0.1b",
933 --     date          = "2020-08-30",
934 --     description   = "Full implementation of kashida feature in XeLaTeX and LuaLaTeX",
935 --     author        = "Hossein Movahhedian",
936 --     copyright     = "Hossein Movahhedian",
937 --     license       = "LPPL v1.3c"
938 -- }
939 --
940 -- -- ^A%% texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
941 -- local err, warn, info, log = luatexbase.provides_module(texnegar_ini.module)
942 -- texnegar_ini.log      = log or (function (s) luatexbase.module_info("texnegar_ini", s) end)
943 -- texnegar_ini.warning = warn or (function (s) luatexbase.module_warning("texnegar_ini", s) end)
944 -- texnegar_ini.error   = err or (function (s) luatexbase.module_error("texnegar_ini", s) end)
945
946 c_true_bool = token.create("c_true_bool")
947

```

```

948 l_texnegar_color_bool = token.create("l_texnegar_color_bool")
949
950 if l_texnegar_color_bool.mode == c_true_bool.mode then
951   color_tbl = color_tbl or {}
952   for item in l_texnegar_color_rgb_tl:gmatch("[^,%s]+") do
953     table.insert(color_tbl, item)
954   end
955 end
956
957 dofile(kpse.find_file("texnegar-luatex-kashida.lua")) -- leaders + resized glyph U+0640
958 --
959 --
960 -- End of file 'texnegar-ini.lua'.
961 </texnegar-ini.lua>

```

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

```

962 <*texnegar-luatex-kashida.lua>
963 --
964 -- This is file 'texnegar-luatex-kashida.lua',
965 -- generated with the docstrip utility.
966 --
967 -- The original source files were:
968 --
969 -- texnegar.dtx (with options: 'texnegar-luatex-kashida-lua')
970 --
971 -- Copyright (C) 2020 Hossein Movahhedian
972 --
973 -- It may be distributed and/or modified under the LaTeX Project Public License,
974 -- version 1.3c or higher (your choice). The latest version of
975 -- this license is at: http://www.latex-project.org/lppl.txt
976 --
977 -- texnegar_luatex_kashida = texnegar_luatex_kashida or {}
978 -- local texnegar_luatex_kashida = texnegar_luatex_kashida
979 -- texnegar_luatex_kashida.module = {
980 --   name = "texnegar_luatex_kashida",
981 --   version = "0.1b",
982 --   date = "2020-08-30",
983 --   description = "Full implementation of kashida feature in XeLaTeX and L",
984 --   author = "Hossein Movahhedian",
985 --   copyright = "Hossein Movahhedian",
986 --   license = "LPPL v1.3c"
987 -- }
988 --
989 -- -- ^A% texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
990 -- local err, warn, info, log = luatexbase.provides_module(texnegar_luatex_kashida.module)
991 -- texnegar_luatex_kashida.log = log or (function (s) luatexbase.module_info("texnegar_
992 -- texnegar_luatex_kashida.warning = warn or (function (s) luatexbase.module_warning("texneg
993 -- texnegar_luatex_kashida.error = err or (function (s) luatexbase.module_error("texnegar
994
995 local peCharTableInitial, peCharTableMedial, peCharTableFinal = dofile(kpse.find_file("texne
char-table.lua"))
996
997 local kashida_unicode = 1600
998 local kashida_subtype = 256

```

```

999
1000 local COLORSTACK = node.subtype("pdf_colorstack")
1001 local node_id      = node.id
1002 local GLUE         = node_id("glue")
1003 local GLYPH        = node_id("glyph")
1004 local HLIST        = node_id("hlist")
1005 local RULE         = node_id("rule")
1006 local WHATSIT      = node_id("whatsit")
1007
1008 local l_texnegar_kashida_glyph_bool      = token.create("l_texnegar_kashida_glyph_bool")
1009 local l_texnegar_kashida_leaders_glyph_bool = token.create("l_texnegar_kashida_leaders_glyph")
1010 local l_texnegar_kashida_leaders_hrulerule_bool = token.create("l_texnegar_kashida_leaders_hrulerule")
1011
1012 local l_texnegar_hboxrecursion_bool      = token.create("l_texnegar_hboxrecursion_bool")
1013 local l_texnegar_vboxrecursion_bool      = token.create("l_texnegar_vboxrecursion_bool")
1014
1015 local selected_font = font.current()
1016 local selected_font_old = selected_font
1017
1018 local string_format = string.format
1019 local debug_getinfo = debug.getinfo
1020
1021 function GetGlyphDimensions(font_file, glyph_index)
1022     local funcName      = debug_getinfo(1).name
1023     local funcNparams   = debug_getinfo(1).nparams
1024
1025     local fnt = fontloader.open(font_file)
1026     local idx = 0
1027     local fnt_glyphcnt = fnt.glyphcnt
1028     local fnt_glyphmin = fnt.glyphmin
1029     local fnt_glyphmax = fnt.glyphmax
1030     if fnt_glyphcnt > 0 then
1031         for idx = fnt_glyphmin, fnt_glyphmax do
1032             local gl = fnt.glyphs[idx]
1033             if gl then
1034                 local gl_unicode = gl.unicode
1035                 if gl_unicode == glyph_index then
1036                     local gl_name      = gl.name
1037                     local gl_width     = gl.width
1038                     local gl_bbox      = gl.boundingBox
1039                     local gl_llx       = gl_bbox[1]
1040                     local gl_depth     = gl_bbox[2]
1041                     local gl_urx       = gl_bbox[3]
1042                     local gl_height    = gl_bbox[4]
1043                     break
1044                 end
1045             end
1046             idx = idx + 1
1047         end
1048     end
1049     fontloader.close(fnt)
1050     return {width = gl_width, height = gl_height, depth = gl_depth, llx = gl_llx, urx = gl_urx}
1051 end
1052

```

```

1053 function GetGlue(t_plb_line_glue_node, t_plb_node)
1054     local funcName      = debug_getinfo(1).name
1055     local funcNparams   = debug_getinfo(1).nparams
1056
1057     local glue_id        = t_plb_line_glue_node.id
1058     local glue_subtype   = t_plb_line_glue_node.subtype
1059     local glue_width     = t_plb_line_glue_node.width
1060     local glue_stretch   = t_plb_line_glue_node.stretch
1061     local glue_shrink    = t_plb_line_glue_node.shrink
1062     local eff_glue_width = node.effective_glue(t_plb_line_glue_node, t_plb_node)
1063     local glue_stretch_order = t_plb_line_glue_node.stretch_order
1064     local glue_shrink_order = t_plb_line_glue_node.shrink_order
1065     local glue_delta     = 0
1066     glue_delta = eff_glue_width - glue_width
1067     return { id = glue_id, subtype = glue_subtype, width = glue_width, stretch = glue_stretch,
1068             shrink = glue_shrink, stretch_order = glue_stretch_order, shrink_order = glue_shrink_order,
1069             effective_glue = eff_glue_width, delta = glue_delta }
1070 end
1071
1072 function GetGlyph(t_plb_line_glyph_node, t_tbl_line_fields, t_CharTableInitial, t_CharTableMedial, t_CharTableFinal)
1073     local funcName      = debug_getinfo(1).name
1074     local funcNparams   = debug_getinfo(1).nparams
1075
1076     local glyph_id      = t_plb_line_glyph_node.id
1077     local glyph_subtype  = t_plb_line_glyph_node.subtype
1078     local glyph_char     = t_plb_line_glyph_node.char
1079     local glyph_font     = t_plb_line_glyph_node.font
1080     local glyph_lang     = t_plb_line_glyph_node.lang
1081     local glyph_width    = t_plb_line_glyph_node.width
1082     local glyph_data     = t_plb_line_glyph_node.data
1083     if not (t_CharTableInitial[glyph_char] == nil) then
1084         t_tbl_line_fields.joinerCharInitial = t_tbl_line_fields.joinerCharInitial + 1
1085         t_plb_line_glyph_node.data = 1
1086     elseif not (t_CharTableMedial[glyph_char] == nil) then
1087         t_tbl_line_fields.joinerCharMedial = t_tbl_line_fields.joinerCharMedial + 1
1088         t_plb_line_glyph_node.data = 2
1089     elseif not (t_CharTableFinal[glyph_char] == nil) then
1090         t_tbl_line_fields.joinerCharFinal = t_tbl_line_fields.joinerCharFinal + 1
1091         t_plb_line_glyph_node.data = 3
1092     end
1093     return { id = glyph_id, subtype = glyph_subtype, char = glyph_char, font = glyph_font, lang = glyph_lang, width = glyph_width, data = glyph_data }
1094 end
1095
1096 function ProcessTableKashidaHlist(ksh_hlistNode, hbox_num, in_font)
1097     local funcName      = debug_getinfo(1).name
1098     local funcNparams   = debug_getinfo(1).nparams
1099
1100     local ksh_hlistNode_id = ksh_hlistNode.id
1101     local ksh_hlistNode_subtype = ksh_hlistNode.subtype
1102
1103     for tn in node.traverse(ksh_hlistNode.head) do
1104         local tn_id = tn.id
1105         local tn_subtype = tn.subtype
1106

```

```

1107     if tn_id == 0 then
1108         for tp in node.traverse(tn.head) do
1109             local tp_id = tp.id
1110             local tp_subtype = tp.subtype
1111             if tp_id == 29 then
1112                 if l_texnegar_color_bool.mode == c_true_bool.mode then
1113                     local col_str      = color_tbl[1] .. " " .. color_tbl[2] .. " " .. c
1114                     local col_str_rg   = col_str .. " rg "
1115                     local col_str_RG   = col_str .. " RG"
1116
1117                     local color_push   = node.new(WHATSIT, COLORSTACK)
1118                     local color_pop    = node.new(WHATSIT, COLORSTACK)
1119                     color_push.stack   = 0
1120                     color_pop.stack    = 0
1121                     color_push.command = 1
1122                     color_pop.command  = 2
1123                     glue_ratio        = .2
1124                     color_push.data    = col_str_rg .. col_str_RG
1125                     color_pop.data     = col_str_rg .. col_str_RG
1126                     tn.head = node.insert_before(tn.list, tn.head, node.copy(color_push))
1127                     tn.head = node.insert_after(tn.list, node.tail(tn.head), node.copy(c
1128                 end
1129
1130                 local tp_font = tp.font
1131                 local tp_char = tp.char
1132                 tp.font = in_font
1133
1134                 local ksh_unicode
1135                 ksh_unicode = font.getfont(in_font).resources.unicodes['kashida']
1136                 if hbox_num == 'l_texnegar_k_box' then
1137                     tp.char = kashida_unicode
1138                 elseif hbox_num == 'l_texnegar_ksh_box' then
1139                     tp.char = ksh_unicode
1140                     tn_width = tn.width
1141                     ksh_hlistNode.width = tn_width
1142                 end
1143             elseif tp_id == 0 then
1144                 if tp.subtype ~= 3 then
1145                     tbl_kashida_hlist_nodes[ #tbl_kashida_hlist_nodes + 1 ] = tp
1146                 end
1147             end
1148         end
1149     elseif tn_id == 1 then
1150         do end
1151     elseif tn_id == 8 then
1152         do end
1153     elseif tn_id == 29 then
1154         if l_texnegar_color_bool.mode == c_true_bool.mode then
1155             local col_str      = color_tbl[1] .. " " .. color_tbl[2] .. " " .. color_tbl
1156             local col_str_rg   = col_str .. " rg "
1157             local col_str_RG   = col_str .. " RG"
1158
1159             local color_push   = node.new(WHATSIT, COLORSTACK)
1160             local color_pop    = node.new(WHATSIT, COLORSTACK)

```



```

1161         color_push.stack = 0
1162         color_pop.stack = 0
1163         color_push.command = 1
1164         color_pop.command = 2
1165         glue_ratio = .2
1166         color_push.data = col_str_rg .. col_str_RG
1167         color_pop.data = col_str_rg .. col_str_RG
1168         ksh_hlistNode.head = node.insert_before(ksh_hlistNode.list, ksh_hlistNode.head)
1169         ksh_hlistNode.head = node.insert_after(ksh_hlistNode.list, node.tail(ksh_hlistNode.list))
1170     end
1171
1172     local tn_font = tn.font
1173     local tn_char = tn.char
1174     tn.font = in_font
1175
1176     local ksh_unicode
1177     ksh_unicode = font.getfont(in_font).resources.unicodes['kashida']
1178     if hbox_num == 'l_texnegar_k_box' then
1179         tn.char = kashida_unicode
1180     elseif hbox_num == 'l_texnegar_ksh_box' then
1181         tn.char = ksh_unicode
1182         tn_width = tn.width
1183         ksh_hlistNode.width = tn_width
1184     end
1185 else
1186     print(string_format("\n tn. Not processed node id is: %d", tn_id))
1187 end
1188 end
1189 end
1190
1191 function SetFontInHbox(hbox_num, font_num)
1192     local funcName = debug_getinfo(1).name
1193     local funcNparams = debug_getinfo(1).nparams
1194
1195     tbl_kashida_hlist_nodes = {}
1196
1197     local tmp_node
1198     tmp_node = node.new("hlist")
1199     tmp_node = tex.getbox(hbox_num)
1200
1201     ProcessTableKashidaHlist(tmp_node, hbox_num, font_num)
1202
1203     ::kashida_hlist_BEGIN::
1204     if #tbl_kashida_hlist_nodes > 0 then
1205         local kashida_hlistNodeAdded = table.remove(tbl_kashida_hlist_nodes,1)
1206         ProcessTableKashidaHlist(kashida_hlistNodeAdded, hbox_num, font_num)
1207         goto kashida_hlist_BEGIN
1208     end
1209 end
1210
1211 function StretchGlyph(t_plb_node, t_plb_glyph_node, t_gluePerJoiner, t_dir, t_filler)
1212     local funcName = debug_getinfo(1).name
1213     local funcNparams = debug_getinfo(1).nparams
1214

```

```

1215     if t_filler == "resized_kashida" then
1216         SetFontInHbox('l_texnegar_k_box', selected_font)
1217     elseif t_filler == "leaders+kashida" then
1218         SetFontInHbox('l_texnegar_ksh_box', selected_font)
1219     end
1220
1221     kashida_node = node.new(GLYPH)
1222     node_glue    = node.new(GLUE)
1223     node_rule    = node.new(RULE)
1224     node_hlist   = node.new(HLIST)
1225
1226     font_current = selected_font
1227     font_name    = font.fonts[font_current].fullname
1228     font_file    = font.fonts[font_current].filename
1229     kashida_char = font.fonts[font_current].characters[1600]
1230
1231     kashida_node.subtype = kashida_subtype
1232     kashida_node.font    = font_current
1233     kashida_node.char    = kashida_unicode
1234     kashida_node.lang    = tex.language
1235
1236     kashida_width  = kashida_node.width
1237     kashida_height = kashida_node.height
1238     kashida_depth  = kashida_node.depth
1239
1240     tbl_gl_dimen = GetGlyphDimensions(font_file, kashida_unicode)
1241     ksh_width, ksh_height, ksh_depth, ksh_llx, ksh_urx =
1242         tbl_gl_dimen.width, tbl_gl_dimen.height, tbl_gl_dimen.depth, tbl_gl_dimen.llx, tbl_g
1243
1244     ratio_width = kashida_width / ksh_width
1245     leaders_height = ratio_width * ksh_height
1246     leaders_depth = - ratio_width * ksh_depth
1247
1248     node_glue.subtype = 100
1249     node.setglue(node_glue, t_gluePerJoiner, 0, 0, 0, 0)
1250
1251     if t_filler == "resized_kashida" then
1252         node_glue.leader = node.copy_list(tex.box['l_texnegar_k_box'])
1253     elseif t_filler == "leaders+kashida" then
1254         node_glue.leader = node.copy_list(tex.box['l_texnegar_ksh_box'])
1255     elseif t_filler == "leaders+hrule" then
1256         node_glue.leader = node_rule
1257     end
1258
1259     node_glue.leader.subtype = 0
1260     node_glue.leader.height  = leaders_height
1261     node_glue.leader.depth   = leaders_depth
1262
1263     node_glue.leader.dir     = t_dir
1264
1265     node.insert_after(t_plb_node.list, t_plb_glyph_node, node_glue)
1266     if t_filler == "leaders+hrule" then
1267         for tn in node.traverse(t_plb_node.head) do
1268             local tn_id = tn.id

```

```

1269         local tn_subtype = tn.subtype
1270
1271         if tn_id == 12 and tn_subtype == 100 then
1272             local t_hbox = node.new(HLIST)
1273             local t_hrulerule = node.copy(tn)
1274             t_hbox.head = node.insert_after(t_hbox.list, t_hbox.head, t_hrulerule)
1275             t_plb_node.head = node.insert_after(t_plb_node.list, tn, t_hbox)
1276
1277             if l_texpnegar_color_bool.mode == c_true_bool.mode then
1278                 local col_str = color_tbl[1] .. " " .. color_tbl[2] .. " " .. color
1279                 local col_str_rg = col_str .. " rg "
1280                 local col_str_RG = col_str .. " RG"
1281
1282                 local color_push = node.new(WHATSIT, COLORSTACK)
1283                 local color_pop = node.new(WHATSIT, COLORSTACK)
1284                 color_push.stack = 0
1285                 color_pop.stack = 0
1286                 color_push.command = 1
1287                 color_pop.command = 2
1288                 glue_ratio = .2
1289                 color_push.data = col_str_rg .. col_str_RG
1290                 color_pop.data = col_str_rg .. col_str_RG
1291                 t_hbox.head = node.insert_before(t_hbox.list, t_hbox.head, node.copy(col
1292                 t_hbox.head = node.insert_after(t_hbox.list, node.tail(t_hbox.head), node
1293             end
1294         end
1295     end
1296 end
1297 end
1298
1299 function GetFillerSpec(t_plb_node, t_plb_head_node, t_tbl_line_fields, t_CharTableInitial, t
1300     local funcName = debug_getinfo(1).name
1301     local funcNparams = debug_getinfo(1).nparams
1302
1303     t_plb_node_id = t_plb_node.id
1304     t_plb_node_subtype = t_plb_node.subtype
1305
1306     for p in node.traverse(t_plb_head_node) do
1307         local p_id = p.id
1308         local p_subtype = p.subtype
1309         if p_id == 0 then
1310             t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - p
1311             if p_subtype ~= 3 then
1312                 tbl_hlist_nodes[ #tbl_hlist_nodes + 1 ] = p
1313             end
1314         elseif p_id == 1 then
1315             t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - p
1316             tbl_vlist_nodes[ #tbl_vlist_nodes + 1 ] = p
1317         elseif p_id == 12 then
1318             tbl_p_glue = GetGlue(p, t_plb_node)
1319             t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - tb
1320             t_tbl_line_fields.total_glues = t_tbl_line_fields.total_glues + 1
1321             t_tbl_line_fields.stretchedGlue = t_tbl_line_fields.stretchedGlue + tbl_p_glue["
1322         elseif p_id == 29 then

```

```

1323         tbl_p_glyph, t_tbl_line_fields = GetGlyph(p, t_tbl_line_fields, t_CharTableIniti
1324         selected_font_old = selected_font
1325         selected_font = tbl_p_glyph["font"]
1326         t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - t
1327         t_tbl_line_fields.total_glyphs = t_tbl_line_fields.total_glyphs + 1
1328     end
1329 end
1330
1331 t_tbl_line_fields.total_joiners = t_tbl_line_fields.joinerCharInitial + t_tbl_line_field
1332 t_tbl_line_fields.gluePerJoiner = 0
1333 if t_tbl_line_fields.total_glues == 0 then
1334     t_tbl_line_fields.stretchedGlue = t_tbl_line_fields.lineWidthRemainder
1335 end
1336 if t_tbl_line_fields.total_joiners > 0 then
1337     t_tbl_line_fields.gluePerJoiner = t_tbl_line_fields.stretchedGlue // t_tbl
1338     t_tbl_line_fields.stretchedGlueRemaineder = t_tbl_line_fields.stretchedGlue % t_tbl
1339 elseif t_tbl_line_fields.total_joiners == 1 then
1340     t_tbl_line_fields.gluePerJoiner = t_tbl_line_fields.stretchedGlue
1341 end
1342
1343 return t_tbl_line_fields
1344 end
1345
1346 function ProcessTableHlist(tmphl_n)
1347     local funcName = debug_getinfo(1).name
1348     local funcNparams = debug_getinfo(1).nparams
1349
1350     local tmphl_n_id = tmphl_n.id
1351     local tmphl_n_subtype = tmphl_n.subtype
1352
1353     local tbl_line_fields = { line_dir = "", line_width = 0, lineWidthRemainder
1354                             joinerCharInitial = 0, joinerCharMedial = 0, joinerCharFinal
1355                             stretchedGlue = 0, total_glues = 0, gluePerJoiner
1356
1357     local tbl_p_glue, tbl_p_glyph
1358
1359     if (tmphl_n_id == 0) and (tmphl_n_subtype == 1 or tmphl_n_subtype == 2) then
1360         tbl_line_fields.line_width = tmphl_n.width
1361         tbl_line_fields.line_dir = tmphl_n.dir
1362         tbl_line_fields.lineWidthRemainder = tbl_line_fields.line_width
1363
1364         if tbl_line_fields.line_dir == "TLT" then
1365             tbl_line_fields = GetFillerSpec(tmphl_n, tmphl_n.head, tbl_line_fields, peCharTa
1366
1367             if tbl_line_fields.total_joiners == 0 or tbl_line_fields.gluePerJoiner == 0 or
1368                 goto continue
1369             end
1370
1371             for q in node.traverse_id(GLUE, tmphl_n.head) do
1372                 local eff_glue_width = node.effective_glue(q, tmphl_n)
1373                 node.setglue(q, q.width, 0, 0, q.stretch_order, q.glue_shrink_order)
1374             end
1375
1376             for r in node.traverse_id(GLYPH, tmphl_n.head) do

```

```

1377         local r_data = r.data
1378         if r_data == 1 or r.data == 2 then
1379             StretchGlyph(tmp_hl_n, r, tbl_line_fields.gluePerJoiner, tbl_line_fields.
1380             elseif r.data == 3 then
1381                 goto for_loop_01
1382             end
1383             ::for_loop_01::
1384         end
1385         tbl_line_fields.line_width = tmp_hl_n.width
1386         tbl_line_fields.lineWidthRemainder = line_width
1387     elseif tbl_line_fields.line_dir == "TRT" then
1388         tbl_line_fields = GetFillerSpec(tmp_hl_n, tmp_hl_n.head, tbl_line_fields, peCharTa
1389         if tbl_line_fields.total_joiners == 0 or tbl_line_fields.gluePerJoiner == 0 or
1390             goto continue
1391         end
1392
1393         for q in node.traverse_id(GLUE, tmp_hl_n.head) do
1394             local eff_glue_width = node.effective_glue(q, tmp_hl_n)
1395             node.setglue(q, q.width, 0, 0, q.stretch_order, q.glue_shrink_order)
1396         end
1397
1398         for r in node.traverse_id(GLYPH, tmp_hl_n.head) do
1399             local r_data = r.data
1400             if r_data == 1 or r.data == 2 then
1401                 StretchGlyph(tmp_hl_n, r, tbl_line_fields.gluePerJoiner, tbl_line_fields.
1402             elseif r.data == 3 then
1403                 goto for_loop_02
1404             end
1405             ::for_loop_02::
1406         end
1407         tbl_line_fields.line_width = tmp_hl_n.width
1408         tbl_line_fields.lineWidthRemainder = line_width
1409     else
1410         print(string_format("\n Line direction '%s' is not supported yet!", tbl_line_fie
1411     end
1412 end
1413 ::continue::
1414 end
1415
1416 function ProcessTableVlist(tmp_vl_n)
1417     local funcName = debug_getinfo(1).name
1418     local funcNparams = debug_getinfo(1).nparams
1419
1420     local tmp_vl_n_id = tmp_vl_n.id
1421     local tmp_vl_n_subtype = tmp_vl_n.subtype
1422
1423     print(string_format(" %s: 00-0 tmp_vl_n: id: %d, subtype: %d", funcName, tmp_vl_n_id, tmpv
1424     for vbNode in node.traverse(tmp_vl_n) do
1425         if vbNode.id == 1 and vbNode.subtype == 0 then
1426             for tr_vbNode in node.traverse(vbNode.head) do
1427                 if (tr_vbNode.id == 0) and (tr_vbNode.subtype == 1 or tr_vbNode.subtype ==
1428                     ProcessTableHlist(tr_vbNode)
1429             end
1430         end
1431     end

```

```

1431         end
1432     end
1433 end
1434
1435 function PostLineBreakFilter(hboxes_stack, groupcode)
1436     local funcName      = debug_getinfo(1).name
1437     local funcNparams = debug_getinfo(1).nparams
1438
1439     tbl_hlist_nodes = {}
1440     tbl_vlist_nodes = {}
1441     for hlistNode in node.traverse(hboxes_stack) do
1442         if node.next(hlistNode) == nil then
1443             goto END
1444         end
1445
1446         ProcessTableHlist(hlistNode)
1447
1448         if l_texpnegar_hboxrecursion_bool.mode == c_true_bool.mode then
1449             ::hboxBEGIN::
1450             if #tbl_hlist_nodes > 0 then
1451                 local hlistNodeAdded = table.remove(tbl_hlist_nodes,1)
1452                 ProcessTableHlist(hlistNodeAdded)
1453                 goto hboxBEGIN
1454             end
1455         end
1456
1457         if l_texpnegar_vboxrecursion_bool.mode == c_true_bool.mode then
1458             ::vboxBEGIN::
1459             if #tbl_vlist_nodes > 0 then
1460                 local vlistNodeAdded = table.remove(tbl_vlist_nodes,1)
1461                 ProcessTableVlist(vlistNodeAdded)
1462                 goto vboxBEGIN
1463             end
1464         end
1465
1466         ::END::
1467     end
1468     return hboxes_stack
1469 end
1470
1471 if l_texpnegar_kashida_glyph_bool.mode == c_true_bool.mode then
1472     filler_pe = "resized_kashida"
1473 elseif l_texpnegar_kashida_leaders_glyph_bool.mode == c_true_bool.mode then
1474     filler_pe = "leaders+kashida"
1475 elseif l_texpnegar_kashida_leaders_hrulerule_bool.mode == c_true_bool.mode then
1476     filler_pe = "leaders+hrulerule"
1477 else
1478     print(string_format" Unknown kashida value.")
1479 end
1480
1481 function StartStretching()
1482     if not luatexbase.in_callback('post_linebreak_filter', 'insertKashida') then
1483         luatexbase.add_to_callback('post_linebreak_filter', PostLineBreakFilter, 'insertKashida')
1484     end

```

```

1485 end
1486
1487 function StopStretching()
1488     if luatexbase.in_callback('post_linebreak_filter', 'insertKashida') then
1489         luatexbase.remove_from_callback('post_linebreak_filter', 'insertKashida')
1490     end
1491 end
1492 --
1493 --
1494 -- End of file 'texnegar-luatex-kashida.lua'.
1495 </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.

## 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 TeX 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 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> Sources*, [CTAN:macros/latex/base/source2e.pdf](#), 2020-02-02
- [9] Package `l3kernel`: The LaTeX3 Team, *The L<sup>A</sup>T<sub>E</sub>X 3 Sources*, [CTAN:macros/latex/contrib/l3kernel/source3.pdf](#), 2020-07-17
- [10] Package `l3kernel`: The LaTeX3 Team, *The L<sup>A</sup>T<sub>E</sub>X 3 Interfaces*, [CTAN:macros/latex/contrib/l3kernel/interface3.pdf](#), 2020-07-17
- [11] Package `luatex`: The LuaTeX Team, *LuaTeX Reference Manual*, [CTAN:systems/doc/luatex/luatex.pdf](#), 2020
- [12] Package `xetexref`: Will Robertson, Khaled Hosny, and Karl Berry, X<sub>Ǝ</sub>T<sub>E</sub>X reference guide, [CTAN:info/xetexref/xetex-reference.pdf](#), 2019-12-09
- [13] Package `xetex`: Jonathan Kew, About X<sub>Ǝ</sub>T<sub>E</sub>X, [CTAN:systems/doc/xetex/XeTeX-notes.pdf](#), 2005-10-17
- [14] Package `xetex`: Michel Goossens, The X<sub>Ǝ</sub>T<sub>E</sub>X Companion, <http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf>, 2009-08-19
- [15] Website: Stack Exchange: Hot Questions, T<sub>E</sub>X-L<sup>A</sup>T<sub>E</sub>X Q&A for users of TeX, LaTeX, ConTeXt, and related typesetting systems, [tex.stackexchange.com](http://tex.stackexchange.com)
- [16] Website: LuaTeX Wiki, LuaT<sub>E</sub>X Wiki, [wiki.luatex.org](http://wiki.luatex.org)

## Index

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

<b>Symbols</b>		<code>\bool_if:nTF</code> ..... 478, 616, 700
<code>\[</code> ..... 613		<code>\bool_set_false:N</code> ..... . 126, 128, 129, 130, 132, 133, 134,
<code>\]</code> ..... 20		135, 136, 292, 401, 407, 418, 424, 440
<code>\_</code> ..... 613		<code>\bool_set_true:N</code> ..... 274, 275, 280, 281, 286, 287, 297, 298, 315, 372, 386, 405, 409, 422, 426, 439
<b>Numbers</b>		box commands:
<code>\0</code> ..... 529		<code>\box_new:N</code> ..... 76, 77
<code>\1</code> ..... 272, 613		
<code>\2</code> ..... 613		
 <code>\_</code> ..... 488, 529, 613		<b>C</b>
<b>A</b>		<code>\c</code> ..... 529
<code>\AtBeginDocument</code> ..... 50, 268		<code>\char</code> ..... 41, 488
<b>B</b>		clist commands:
bool commands:		<code>\clist_count:N</code> ..... 471
<code>\bool_if:NTF</code> . 33, 46, 64, 446, 467, 492, 547, 561, 606, 680, 685, 706, 711		<code>\clist_item:Nn</code> ..... 476, 477
		<code>\clist_map_inline:Nn</code> ..... ..... 483, 649, 655, 661, 667, 673
		<code>\clist_new:N</code> ..... 469



<code>\clist_put_left:Nn</code> . . . . .	480	<code>\int_const:Nn</code> . . . . .	82, 83, 84, 86, 87
<code>\clist_set:Nn</code> . . . . .	199,	<code>\int_eval:n</code> . . . . .	475
200, 201, 202, 203, 204, 205, 206,		<code>\int_new:N</code> . . . . .	92,
207, 209, 477, 648, 654, 660, 666, 672		94, 96, 98, 99, 100, 101, 102, 104, 470	
<code>\l_tmpa_clist</code> . . . . .	477, 480	<code>\int_set:Nn</code> . . . . .	138,
<code>\color</code> . . . . .	554	139, 140, 141, 142, 306, 309, 310,	
<code>\colorlet</code> . . . . .	549, 552	311, 312, 313, 314, 471, 474, 475, 520	
<code>\convertcolspec</code> . . . . .	389	<code>\int_step_inline:nnnn</code> . . . . .	472
<code>\copy</code> . . . . .	620	<code>\int_use:N</code> . . . . .	638, 639
cs commands:		<code>\c_one_int</code> . . . . .	565
<code>\cs:w</code> . . . . .	575, 585, 591, 595	<code>\l_tmpa_int</code> . . . . .	306, 307, 474, 475, 476
<code>\cs_end:</code> . . . . .	575, 585, 591, 595	<code>\l_tmpb_int</code> . . . . .	475, 477
<code>\cs_new:Nn</code> . . . . .	495	iow commands:	
<code>\cs_new:Npn</code> . . . . .	559, 604	<code>\iow_now:Nn</code> . . . . .	573, 583
<code>\cs_new_protected:Nn</code> . . . . .	526		
		<b>K</b>	
<b>D</b>		<code>\KashidaHMFixOff</code> . . . . .	433, 440, 444
<code>\def</code> . . . . .	74	<code>\KashidaHMFixOn</code> . . . . .	434, 439, 443
dim commands:		<code>\KashidaOff</code> . . . . .	444
<code>\dim_compare:nTF</code> . . . . .	512, 597	<code>\KashidaOn</code> . . . . .	52, 443
<code>\dim_eval:n</code> . . . . .	595	keys commands:	
<code>\dim_new:N</code> . . . . .	124	<code>\keys_define:nn</code> . . . . .	257
<code>\dim_set:Nn</code> . . . . .	593		
<code>\directlua</code> . . . . .	43, 390, 433, 434	<b>L</b>	
<code>\discouragebadlinebreaks</code> . . . . .	517	<code>\llap</code> . . . . .	599
		lua commands:	
<b>E</b>		<code>\lua_now:n</code> . . . . .	41
else commands:		<code>\luatexversion</code> . . . . .	35
<code>\else:</code> . . . . .	508, 592		
<code>\endinput</code> . . . . .	11, 16, 26, 56, 69, 462, 730	<b>M</b>	
exp commands:		<code>\makeatletter</code> . . . . .	454
<code>\exp_after:wN</code> . . . . .	575, 585, 589	<code>\makeatother</code> . . . . .	460
		<code>\MessageBreak</code> . . . . .	504, 506
<b>F</b>		mode commands:	
fi commands:		<code>\mode_leave_vertical:</code> . . . . .	564
<code>\fi:</code> . . . . .	37, 514, 600	msg commands:	
		<code>\msg_error:nn</code> . . . . .	301
<b>G</b>		<code>\msg_error:nnn</code> . . . . .	624, 726
<code>\gdef</code> . . . . .	575, 585	<code>\msg_error:nnnnn</code> . . . . .	36
<code>\GenericError</code> . . . . .	497	<code>\msg_fatal:nn</code> . . . . .	24
		<code>\msg_new:nnn</code> . . . . .	
<b>H</b>		18, 221, 226, 231, 236, 242, 252	
hbox commands:		<code>\msg_warning:nnn</code> . . . . .	266
<code>\hbox_set:Nn</code> . . . . .	39, 41, 618		
<code>\height</code> . . . . .	39, 599	<b>N</b>	
		Negar commands:	
<b>I</b>		<code>Negar:</code> . . . . .	1
if commands:		<code>\NewDocumentCommand</code> . . . . .	433, 434, 439, 440, 517
<code>\if_int_compare:w</code> . . . . .	35	<code>\newif</code> . . . . .	455, 459
<code>\if_meaning:w</code> . . . . .	590	<code>\newXeTeXintercharclass</code> . . . . .	
<code>\if_mode_vertical:</code> . . . . .	496	537, 538, 539, 540, 541	
<code>\IfNoValueF</code> . . . . .	519, 521		
int commands:		<b>P</b>	
<code>\l_fontnumber_int</code> . . . . .	104	<code>\ProcessKeysOptions</code> . . . . .	429
<code>\int_case:nnTF</code> . . . . .	307	<code>\ProvidesExplFile</code> . . . . .	72, 465, 535
		<code>\ProvidesExplPackage</code> . . . . .	6, 29, 60

<b>R</b>	
regex commands:	
\regex_replace_all:nnN . . . . .	529
\regex_replace_once:nnN . . . . .	272, 613
\relax . . . . .	613
\RequirePackage . . . . .	2, 3, 4, 5, 59
\RequirePackageWithOptions . . . . .	10, 15
\resizebox . . . . .	39, 599
<b>S</b>	
seq commands:	
\seq_pop_left:NN . . . . .	486, 487
\seq_set_split:Nnn . . . . .	485
\l_tmpa_seq . . . . .	485, 486, 487
skip commands:	
\skip_horizontal:N . . . . .	513
\skip_horizontal:n . . . . .	579, 620, 639
\l_tmpa_skip . . . . .	509, 512, 513
\c_zero_skip . . . . .	512
\space . . . . .	498, 505, 507
str commands:	
\str_if_eq_p:NN . . . . .	616
sys commands:	
\sys_if_engine luatex:TF . . . . .	8, 387, 431
\sys_if_engine xetex:TF . . . . .	13, 437
<b>T</b>	
\TeX . . . . .	74
TeX and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
\if@Kashida@on . . . . .	455
\if@Kashida@XB@fix . . . . .	459
tex commands:	
\tex_advance:D . . . . .	565
\tex_font:D . . . . .	270, 610
\tex_fontname:D . . . . .	270, 610
\tex_global:D . . . . .	565
\tex_hrule:D . . . . .	637
\tex_ignorespaces:D . . . . .	513
\tex_input:D . . . . .	31, 48, 62, 66, 543
\tex_lastskip:D . . . . .	509
\tex_leaders:D . . . . .	620, 637
\tex_let:D . . . . .	443, 444
\tex_penalty:D . . . . .	511, 563, 619, 636
\tex_relax:D . . . . .	591, 599
\tex_romannumeral:D . . . . .	567, 568
\tex_the:D . . . . .	270, 610
\tex_unskip:D . . . . .	510
\TeXNegar . . . . .	74
texnegar commands:	
\c_texnegar_a_charclass . . . . .	540, 651, 695, 697, 721, 723
\l_texnegar_a_clist . . . . .	648, 649
\l_texnegar_active_ligs_tl . . . . .	116, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 374, 478
\l_texnegar_col_default_tl . . . . .	197, 381
\l_texnegar_color_bool . . . . .	136, 386, 547
\l_texnegar_color_rgb_tl . . . . .	122, 389, 390
\l_texnegar_color_tl . . . . .	121, 381, 384, 389, 549
\l_texnegar_counter_int . . . . .	92, 565, 567, 568
\c_texnegar_d_charclass . . . . .	537, 657, 684, 689, 690, 691, 692, 694, 695, 710, 715, 716, 717, 718, 720, 721
\l_texnegar_d_clist . . . . .	654, 655
\l_texnegar_diff_pos_dim . . . . .	124, 593, 597, 599
\l_texnegar_fnt_default_tl . . . . .	184, 351
\l_texnegar_fnt_kayhan_tl . . . . .	156, 323
\l_texnegar_fnt_kayhannavaar_tl . . . . .	157, 324
\l_texnegar_fnt_kayhanpook_tl . . . . .	158, 325
\l_texnegar_fnt_kayhansayeh_tl . . . . .	159, 326
\l_texnegar_fnt_khoramshahr_tl . . . . .	160, 327
\l_texnegar_fnt_khorramshahr_tl . . . . .	161, 328
\l_texnegar_fnt_niloofer_tl . . . . .	162, 329
\l_texnegar_fnt_noskip_tl . . . . .	185, 352
\l_texnegar_fnt_paatch_tl . . . . .	163, 330
\l_texnegar_fnt_riyaz_tl . . . . .	164, 331
\l_texnegar_fnt_roya_tl . . . . .	165, 332
\l_texnegar_fnt_shafigh_tl . . . . .	166, 333
\l_texnegar_fnt_shafighKurd_tl . . . . .	167, 334
\l_texnegar_fnt_shafighUzbek_tl . . . . .	168, 335
\l_texnegar_fnt_shiraz_tl . . . . .	169, 336
\l_texnegar_fnt_sols_tl . . . . .	170, 337
\l_texnegar_fnt_tabriz_tl . . . . .	171, 338
\l_texnegar_fnt_titr_tl . . . . .	172, 339
\l_texnegar_fnt_titre_tl . . . . .	173, 340
\l_texnegar_fnt_traffic_tl . . . . .	174, 341
\l_texnegar_fnt_vahid_tl . . . . .	175, 342
\l_texnegar_fnt_vosta_tl . . . . .	176, 343
\l_texnegar_fnt_yaghut_tl . . . . .	177, 344
\l_texnegar_fnt_yagut_tl . . . . .	178, 345
\l_texnegar_fnt_yas_tl . . . . .	179, 346
\l_texnegar_fnt_yekan_tl . . . . .	180, 347
\l_texnegar_fnt_yermook_tl . . . . .	181, 348
\l_texnegar_fnt_zar_tl . . . . .	182, 349
\l_texnegar_fnt_ziba_tl . . . . .	183, 350
\l_texnegar_font_full_tl . . . . .	111, 610, 611

<code>\l_texnegar_font_init_tl</code>	612, 613, 616	<code>\l_texnegar_lig_medi_clist</code>	205, 217
<code>\l_texnegar_font_name_tl</code>	112, 611, 612, 624	<code>\l_texnegar_lig_medi_tl</code>	193, 217, 368
<code>\c_texnegar_four_int</code>	87, 638	<code>\l_texnegar_lig_names_clist</code>	209, 471, 476, 477
<code>\l_texnegar_gap_filler_tl</code>	118, 267, 279, 285, 291, 296, 300, 301, 608, 678, 701, 702, 726	<code>\l_texnegar_lig_names_len_int</code>	470, 471, 472
<code>\l_texnegar_hboxrecursion_bool</code>	134, 401, 405, 407, 409	<code>\l_texnegar_lig_rlig_clist</code>	206, 218
<code>\l_texnegar_hboxrecursion_off_tl</code>	150, 399	<code>\l_texnegar_lig_rlig_tl</code>	194, 218, 369
<code>\l_texnegar_hboxrecursion_on_tl</code>	151, 403	<code>\l_texnegar_ligature_bool</code>	132, 372, 467
<code>\l_texnegar_high_penalty_int</code>	101, 141, 312	<code>\l_texnegar_ligatures_clist</code>	469, 480, 483
<code>\l_texnegar_k_box</code>	39, 76	<code>\texnegar_line_break:</code>	495
<code>\l_texnegar_kashida_fix_bool</code>	33, 46, 64, 126, 274, 280, 286, 292, 297, 439, 440, 446, 561, 606, 680, 685, 706, 711	<code>\l_texnegar_line_break_penalty_int</code>	96, 309, 310, 311, 312, 313, 314, 511, 517, 520
<code>\texnegar_kashida_glyph</code>	559, 681, 682, 686, 687, 689, 690, 691, 692, 693, 694, 695, 696	<code>\l_texnegar_line_break_tl</code>	106, 528, 529, 530
<code>\l_texnegar_kashida_glyph_bool</code>	128, 275	<code>\l_texnegar_linebreakpenalty_bool</code>	133, 315, 492
<code>\texnegar_kashida_leaders</code>	604, 707, 708, 712, 713, 715, 716, 717, 718, 719, 720, 721, 722	<code>\l_texnegar_low_penalty_int</code>	99, 139, 310
<code>\l_texnegar_kashida_leaders_glyph_bool</code>	129, 281, 298	<code>\c_texnegar_lrm_int</code>	83, 563, 635
<code>\l_texnegar_kashida_leaders_hrulerule_bool</code>	130, 287	<code>\c_texnegar luatexversionmajormin_int</code>	35, 36, 79
<code>\l_texnegar_kashida_slot_int</code>	94	<code>\c_texnegar luatexversionminormin_int</code>	35, 36, 80
<code>\l_texnegar_ksh_box</code>	41, 77, 618, 620	<code>\l_texnegar_main_font_full_tl</code>	108, 270, 271
<code>\c_texnegar_ksh_int</code>	82, 599, 638, 639	<code>\l_texnegar_main_font_name_tl</code>	109, 271, 272
<code>\c_texnegar_l_charclass</code>	538, 663, 691, 692, 693, 696, 697, 717, 718, 719, 722, 723	<code>\l_texnegar_max_penalty_int</code>	102, 142, 313
<code>\l_texnegar_l_clist</code>	660, 661	<code>\l_texnegar_med_penalty_int</code>	100, 140, 311
<code>\l_texnegar_lig_aalt_clist</code>	199, 211	<code>\l_texnegar_min_penalty_int</code>	98, 138, 309
<code>\l_texnegar_lig_aalt_tl</code>	187, 211, 362	<code>\l_texnegar_pos_tl</code>	567, 575, 585, 591, 595
<code>\l_texnegar_lig_ccmp_clist</code>	200, 212	<code>\texnegar_put_line_breaks:n</code>	523, 526
<code>\l_texnegar_lig_ccmp_tl</code>	188, 212, 363	<code>\c_texnegar_r_charclass</code>	539, 669, 694, 696, 720, 722
<code>\l_texnegar_lig_default_clist</code>	207	<code>\l_texnegar_r_clist</code>	666, 667
<code>\l_texnegar_lig_default_tl</code>	195, 370, 374, 478	<code>\c_texnegar_skip_a_tl</code>	89, 451, 682, 687, 689, 690, 691, 692, 693, 694, 695, 696, 708, 713, 715, 716, 717, 718, 719, 720, 721, 722
<code>\l_texnegar_lig_dlig_clist</code>	201, 213	<code>\c_texnegar_skip_b_tl</code>	90, 517
<code>\l_texnegar_lig_dlig_tl</code>	189, 213, 364	<code>\l_texnegar_skip_default_tl</code>	114, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346,
<code>\l_texnegar_lig_fina_clist</code>	202, 214		
<code>\l_texnegar_lig_fina_tl</code>	190, 214, 365		
<code>\l_texnegar_lig_init_clist</code>	203, 215		
<code>\l_texnegar_lig_init_tl</code>	191, 215, 366		
<code>\l_texnegar_lig_locl_clist</code>	204, 216		
<code>\l_texnegar_lig_locl_tl</code>	192, 216, 367		

[illegible]