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

Hossein Movahhedian*

Released 2021-02-09 v0.1e

`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

1.1 File: texnegar.sty

```
⟨*texnegar-sty⟩
\RequirePackage{xparse}
\RequirePackage{l3keys2e}
\RequirePackage{graphicx}[2019-11-30]
\RequirePackage{array}[2019-10-01]
\RequirePackage[dvipsnames,svgnames,x11names]{xcolor}[2016/05/11]
\RequirePackage{fontspec}[2020/02/21]
\RequirePackage{newverbs}[2010/09/02]
\RequirePackage{environ}[2014/05/04]

\ProvidesExplPackage {texnegar} {2021-02-09} {0.1e} { Full implementation of kashida feature

\sys_if_engine_luatex:T
  {
    \RequirePackageWithOptions{texnegar-luatex}
    \endinput
  }
\sys_if_engine_xetex:T
  {
    \RequirePackageWithOptions{texnegar-xetex}
    \endinput
  }
\msg_new:nnn {texnegar} {cannot-use-pdftex}
  {
    The~ texnegar~ package~ requires~ either~ XeTeX~ or~ LuaTeX.\\\\
    You~ must~ change~ your~ typesetting~ engine~ to,~ e.g.,~
    "xelatex"~ or~ "lualatex" instead~ of~ "latex"~ or~ "pdflatex".
  }
\msg_fatal:nn {texnegar} {cannot-use-pdftex}

 \endinput
⟨/texnegar-sty⟩
```

1.2 File: texnegar-luatex.sty

```
⟨*texnegar-luatex-sty⟩
\ProvidesExplPackage {texnegar-luatex} {2021-02-09} {0.1e} { Full implementation of kashida

\tex_input:D { texnegar-ini.tex }

\bool_if:NT \l_texnegar_kashida_fix_bool
  {
    \if_int_compare:w \luatexversion < \c_texnegar_luatexversionmajormin_int\c_texnegar_luat
        \msg_error:nnxxx { texnegar } { luatex-version-is-too-old } { !!!! } { \c_texnegar_l
    \fi:

    \hbox_set:Nn \l_texnegar_k_box { \resizebox{5000sp}{\height}{-} }

    \hbox_set:Nn \l_texnegar_ksh_box { \char\lua_now:n { tex.sprint(0, font.getfont(font.cur

    \directlua{dofile(kpse.find_file("texnegar.lua"))}
  }

\bool_if:NT \l_texnegar_kashida_fix_bool
  {
    \tex_input:D { texnegar-common-kashida.tex }

    \AtBeginDocument
      {
        \KashidaOn
      }
  }

 \endinput
⟨/texnegar-luatex-sty⟩
```

1.3 File: texnegar-xetex.sty

```
⟨*texnegar-xetex-sty⟩
\RequirePackage{zref-savepos}[2020-03-03]
\ProvidesExplPackage {texnegar-xetex} {2021-02-09} {0.1e} { Full implementation of kashida f

\tex_input:D { texnegar-ini.tex }

\bool_if:NT \l_texnegar_kashida_fix_bool
  {
    \tex_input:D { texnegar-xetex-kashida.tex }
  }

 \endinput
⟨/texnegar-xetex-sty⟩
```

1.4 File: texnegar-ini.tex

```
⟨*texnegar-ini-tex⟩
\ProvidesExplFile {texnegar-ini.tex} {2021-02-09} {0.1e} { Full implementation of kashida fe

\def\TeXNegar{\TeX Negar}

\box_new:N \l_texnegar_k_box
\box_new:N \l_texnegar_ksh_box

\tl_const:Nn \c_texnegar_luatexversionmajormin_int {1}
\tl_const:Nn \c_texnegar_luatexversionminormin_int {12}

\int_const:Nn \c_texnegar_ksh_int {"0640} % kashida
\int_const:Nn \c_texnegar_lrm_int {"200E} % left-right-mark
\int_const:Nn \c_texnegar_zwj_int {"200D} % zero-width joiner

\int_const:Nn \c_texnegar_two_int {2}
\int_const:Nn \c_texnegar_four_int {4}

\tl_const:Nn \c_texnegar_skip_a_tl { 0 em plus 0.5 em }
\tl_const:Nn \c_texnegar_skip_b_tl { 0.14 em plus 5.5 em }

\int_new:N \l_texnegar_counter_int

```

```
\int_new:N \l_texnegar_kashida_slot_int

\int_new:N \l_texnegar_line_break_penalty_int

\int_new:N \l_texnegar_min_penalty_int
\int_new:N \l_texnegar_low_penalty_int
\int_new:N \l_texnegar_med_penalty_int
\int_new:N \l_texnegar_high_penalty_int
\int_new:N \l_texnegar_max_penalty_int

\int_new:N \l_fontnumber_int

\tl_new:N \l_texnegar_line_break_tl

\tl_new:N \l_texnegar_main_font_full_tl
\tl_new:N \l_texnegar_main_font_name_tl

\tl_new:N \l_texnegar_font_full_tl
\tl_new:N \l_texnegar_font_name_tl

\tl_new:N \l_texnegar_skip_default_tl

\tl_new:N \l_texnegar_active_ligs_tl

\tl_new:N \l_texnegar_gap_filler_tl

\tl_new:N \l_texnegar_use_color_tl
\tl_new:N \l_texnegar_color_tl
\tl_new:N \l_texnegar_color_rgb_tl

\dim_new:N \l_texnegar_diff_pos_dim

\bool_set_false:N \l_texnegar_minimal_bool
\tl_set:Nn \l_texnegar_minimal_off_tl { Off }
\tl_set:Nn \l_texnegar_minimal_on_tl { On }

\bool_set_false:N \l_texnegar_kashida_fix_bool

\bool_set_false:N \l_texnegar_kashida_fontfamily_bool
\tl_new:N \l_texnegar_kashida_fontfamily_tl
\tl_set:Nn \l_texnegar_kashida_fontfamily_tl { N/A }

\bool_set_false:N \l_texnegar_kashida_glyph_bool
\bool_set_false:N \l_texnegar_kashida_leaders_glyph_bool
\bool_set_false:N \l_texnegar_kashida_leaders_hrule_bool

\bool_set_false:N \l_texnegar_ligature_bool
\bool_set_false:N \l_texnegar_linebreakpenalty_bool
\bool_set_false:N \l_texnegar_hboxrecursion_bool
\bool_set_false:N \l_texnegar_vboxrecursion_bool
\bool_set_false:N \l_texnegar_color_bool

\int_set:Nn \l_texnegar_min_penalty_int { 0 }
\int_set:Nn \l_texnegar_low_penalty_int { 8 }
```

```
\int_set:Nn \l_texnegar_med_penalty_int { 15 }
\int_set:Nn \l_texnegar_high_penalty_int { 25 }
\int_set:Nn \l_texnegar_max_penalty_int { 10000 }

\tl_set:Nn \l_texnegar_stretch_glyph_tl { glyph }
\tl_set:Nn \l_texnegar_stretch_leaders_glyph_tl { leaders+glyph }
\tl_set:Nn \l_texnegar_stretch_leaders_hrule_tl { leaders+hrule }
\tl_set:Nn \l_texnegar_stretch_off_tl { Off }
\tl_set:Nn \l_texnegar_stretch_on_tl { On }

\tl_set:Nn \l_texnegar_hboxrecursion_off_tl { Off }
\tl_set:Nn \l_texnegar_hboxrecursion_on_tl { On }

\tl_set:Nn \l_texnegar_vboxrecursion_off_tl { Off }
\tl_set:Nn \l_texnegar_vboxrecursion_on_tl { On }

\tl_set:Nn \l_texnegar_fnt_kayhan_tl       { kayhan }
\tl_set:Nn \l_texnegar_fnt_kayhannavaar_tl { kayhannavaar }
\tl_set:Nn \l_texnegar_fnt_kayhanpook_tl   { kayhanpook }
\tl_set:Nn \l_texnegar_fnt_kayhansayeh_tl  { kayhansayeh }
\tl_set:Nn \l_texnegar_fnt_khoramshahr_tl  { khoramshahr }
\tl_set:Nn \l_texnegar_fnt_khorramshahr_tl { khorramshahr }
\tl_set:Nn \l_texnegar_fnt_niloofar_tl     { niloofar }
\tl_set:Nn \l_texnegar_fnt_paatch_tl       { paatch }
\tl_set:Nn \l_texnegar_fnt_riyaz_tl        { riyaz }
\tl_set:Nn \l_texnegar_fnt_roya_tl         { roya }
\tl_set:Nn \l_texnegar_fnt_shafigh_tl      { shafigh }
\tl_set:Nn \l_texnegar_fnt_shafighKurd_tl  { shafighKurd }
\tl_set:Nn \l_texnegar_fnt_shafighUzbek_tl { shafighUzbek }
\tl_set:Nn \l_texnegar_fnt_shiraz_tl       { shiraz }
\tl_set:Nn \l_texnegar_fnt_sols_tl         { sols }
\tl_set:Nn \l_texnegar_fnt_tabriz_tl       { tabriz }
\tl_set:Nn \l_texnegar_fnt_titr_tl         { titr }
\tl_set:Nn \l_texnegar_fnt_titre_tl        { titre }
\tl_set:Nn \l_texnegar_fnt_traffic_tl      { traffic }
\tl_set:Nn \l_texnegar_fnt_vahid_tl        { vahid }
\tl_set:Nn \l_texnegar_fnt_vosta_tl        { vosta }
\tl_set:Nn \l_texnegar_fnt_yaghut_tl       { yaghut }
\tl_set:Nn \l_texnegar_fnt_yagut_tl        { yagut }
\tl_set:Nn \l_texnegar_fnt_yas_tl          { yas }
\tl_set:Nn \l_texnegar_fnt_yekan_tl        { yekan }
\tl_set:Nn \l_texnegar_fnt_yermook_tl      { yermook }
\tl_set:Nn \l_texnegar_fnt_zar_tl          { zar }
\tl_set:Nn \l_texnegar_fnt_ziba_tl         { ziba }
\tl_set:Nn \l_texnegar_fnt_default_tl      { default }
\tl_set:Nn \l_texnegar_fnt_noskip_tl       { noskip }

\tl_set:Nn \l_texnegar_lig_aalt_tl    { aalt } % Access All Alternatives
\tl_set:Nn \l_texnegar_lig_ccmp_tl    { ccmp } % Glyph Composition/Decomposition
\tl_set:Nn \l_texnegar_lig_dlig_tl    { dlig } % Discretionary Ligatures
\tl_set:Nn \l_texnegar_lig_fina_tl    { fina } % Final (Terminal) Forms
\tl_set:Nn \l_texnegar_lig_init_tl    { init } % Initial Forms
\tl_set:Nn \l_texnegar_lig_locl_tl    { locl } % Localized Forms
\tl_set:Nn \l_texnegar_lig_medi_tl    { medi } % Medial Forms
```

```
\tl_set:Nn \l_texnegar_lig_rlig_tl    { rlig } % Required Ligatures
\tl_set:Nn \l_texnegar_lig_default_tl { default }

\tl_set:Nn \l_texnegar_col_default_tl { magenta }

\clist_set:Nn \l_texnegar_lig_aalt_clist    { } % Access All Alternatives
\clist_set:Nn \l_texnegar_lig_ccmp_clist    { } % Glyph Composition/Decomposition
\clist_set:Nn \l_texnegar_lig_dlig_clist    { FDF2 =  , FDF3 =  , FDFB =  } % Discretionary
\clist_set:Nn \l_texnegar_lig_fina_clist    { } % Final (Terminal) Forms
\clist_set:Nn \l_texnegar_lig_init_clist    { } % Initial Forms
\clist_set:Nn \l_texnegar_lig_locl_clist    { } % Localized Forms
\clist_set:Nn \l_texnegar_lig_medi_clist    { } % Medial Forms
\clist_set:Nn \l_texnegar_lig_rlig_clist    { } % Required Ligatures
\clist_set:Nn \l_texnegar_lig_default_clist { }

\clist_set:Nn \l_texnegar_lig_names_clist
  {
    \l_texnegar_lig_aalt_tl , { \l_texnegar_lig_aalt_clist } ,
    \l_texnegar_lig_ccmp_tl , { \l_texnegar_lig_ccmp_clist } ,
    \l_texnegar_lig_dlig_tl , { \l_texnegar_lig_dlig_clist } ,
    \l_texnegar_lig_fina_tl , { \l_texnegar_lig_fina_clist } ,
    \l_texnegar_lig_init_tl , { \l_texnegar_lig_init_clist } ,
    \l_texnegar_lig_locl_tl , { \l_texnegar_lig_locl_clist } ,
    \l_texnegar_lig_medi_tl , { \l_texnegar_lig_medi_clist } ,
    \l_texnegar_lig_rlig_tl , { \l_texnegar_lig_rlig_clist } ,
  }

\msg_new:nnn { texnegar } { error-kashida-character-is-not-available-in-the-main-font }
  {
    Sorry,~ kashida~ character~ is~ not~ available~ in~ the~ main~ font~#1!
  }

\msg_new:nnn { texnegar } { error-value-not-available-for-kashida-option }
  {
    Sorry,~ value~ ‘#1’~ is~ not~ available~ for~ ‘Kashida’~ option~ yet~!
  }

\msg_new:nnn { texnegar } { error-specify-value-for-kashida-option }
  {
    Sorry,~ you~ must~ specify~ a~ value~ for~ ‘Kashida’~ option~ yet~!
  }

\msg_new:nnn { texnegar } { warning-experimental-feature }
  {
    Please~ note~ that~ the~ feature~ ‘#1’~ is~ still~ experimental~
    and~ is~ not~ regarded~ as~ stable.
  }

\msg_new:nnn { texnegar } { hm-series-font-not-found }
  {
    Either~ the~ font~‘#1’~ is~ not~ installed~ on~ your~ system~ or~ does~ not~
    belong~ to~ HM~Series~fonts.~
    Please~ note~ that~ the~ option~ ‘Kashida=leaders+glyph’~ is~ currently~ only~
    supported~ by~ HM~Series~fonts.~
    If~ you~ know~ of~ any~ other~ font~ that~ supports~ this~ option,~ please~
    let~ me~ know~ to~ add~ it~ to~ the~ list~ of~ corresponding~ fonts.~
  }

\msg_new:nnn { texnegar } { luatex-version-is-too-old }
  {
    #1:~Your~luatex~is~too~old,~you~need~at~least~version~#2.#3~!
  }

\keys_define:nn { texnegar }
  {
    Kashidafontfamily .code:n =
      {
        \tl_set:Nn \l_tmpa_tl { #1 }
        \tl_case:Nn \l_tmpa_tl
          {
            \tl_if_empty:NTF \l_tmpa_tl
              {
                \bool_set_false:N \l_texnegar_kashida_fontfamily_bool
              }
              {
                \bool_set_true:N \l_texnegar_kashida_fontfamily_bool
                \tl_set:Nx \l_texnegar_kashida_fontfamily_tl { \l_tmpa_tl }
              }
          }
      } ,

    Minimal .code:n =
      {
        \tl_set:Nn \l_tmpa_tl { #1 }
        \tl_case:Nn \l_tmpa_tl
          {
            \l_texnegar_minimal_off_tl
              {
                \bool_set_false:N \l_texnegar_minimal_bool
              }
            \l_texnegar_minimal_on_tl
              {
                \bool_set_true:N \l_texnegar_minimal_bool
              }
          }
      } ,

    Kashida .code:n =
      {
        \tl_set:Nn \l_tmpa_tl { #1 }
        \tl_case:NnTF \l_tmpa_tl
          {
            \l_texnegar_stretch_glyph_tl
              {
                \msg_warning:nnn { texnegar } { warning-experimental-feature } { Kashida=gly
                \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_glyph_tl }
                \AtBeginDocument
```

```
           {
            \tl_set:Nx \l_texnegar_main_font_full_tl { \tex_fontname:D \tex_the:D \t
            \tl_set:Nx \l_texnegar_main_font_name_tl { \l_texnegar_main_font_full_tl
            \regex_replace_once:nnN { ^"([^/]+)/.* } { \1 } \l_texnegar_main_font_na
           }
          \bool_set_true:N \l_texnegar_kashida_fix_bool
          \bool_set_true:N \l_texnegar_kashida_glyph_bool
         }
        \l_texnegar_stretch_leaders_glyph_tl
         {
          \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_leaders_glyph_tl
          \bool_set_true:N \l_texnegar_kashida_fix_bool
          \bool_set_true:N \l_texnegar_kashida_leaders_glyph_bool
         }
        \l_texnegar_stretch_leaders_hrule_tl
         {
          \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_leaders_hrule_tl
          \bool_set_true:N \l_texnegar_kashida_fix_bool
          \bool_set_true:N \l_texnegar_kashida_leaders_hrule_bool
         }
        \l_texnegar_stretch_off_tl
         {
          \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_off_tl }
          \bool_set_false:N \l_texnegar_kashida_fix_bool
         }
        \l_texnegar_stretch_on_tl
         {
          \tl_set:Nx \l_texnegar_gap_filler_tl { \l_texnegar_stretch_leaders_glyph_tl
          \bool_set_true:N \l_texnegar_kashida_fix_bool
          \bool_set_true:N \l_texnegar_kashida_leaders_glyph_bool
         }
       } { } { \tl_set:Nx \l_texnegar_gap_filler_tl { #1 } }
      \tl_if_empty:NT \l_texnegar_gap_filler_tl { \msg_error:nn { texnegar } { error-specify-value-for-kashida-option } }
     } ,

    linebreakpenalty .code:n =
     {
      \int_set:Nn \l_tmpa_int { #1 }
      \int_case:nnTF \l_tmpa_int
       {
        \l_texnegar_min_penalty_int  { \int_set:Nn \l_texnegar_line_break_penalty_int {
        \l_texnegar_low_penalty_int  { \int_set:Nn \l_texnegar_line_break_penalty_int {
        \l_texnegar_med_penalty_int  { \int_set:Nn \l_texnegar_line_break_penalty_int {
        \l_texnegar_high_penalty_int { \int_set:Nn \l_texnegar_line_break_penalty_int {
        \l_texnegar_max_penalty_int  { \int_set:Nn \l_texnegar_line_break_penalty_int {
       } { } { \int_set:Nn \l_texnegar_line_break_penalty_int { #1 } }
      \bool_set_true:N \l_texnegar_linebreakpenalty_bool
     } ,

    kashidastretch .code:n =
     {
      \tl_set:Nn \l_tmpa_tl { #1 }
      \tl_case:NnTF \l_tmpa_tl
      {
         \l_texnegar_fnt_kayhan_tl       { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
         \l_texnegar_fnt_kayhannavaar_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
         \l_texnegar_fnt_kayhanpook_tl   { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
         \l_texnegar_fnt_kayhansayeh_tl  { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
         \l_texnegar_fnt_khoramshahr_tl  { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
         \l_texnegar_fnt_khorramshahr_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
         \l_texnegar_fnt_niloofar_tl     { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
         \l_texnegar_fnt_paatch_tl       { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
         \l_texnegar_fnt_riyaz_tl        { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
         \l_texnegar_fnt_roya_tl         { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
         \l_texnegar_fnt_shafigh_tl      { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
         \l_texnegar_fnt_shafighKurd_tl  { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
         \l_texnegar_fnt_shafighUzbek_tl { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
         \l_texnegar_fnt_shiraz_tl       { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
         \l_texnegar_fnt_sols_tl         { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
         \l_texnegar_fnt_tabriz_tl       { \tl_set:Nn \l_texnegar_skip_default_tl { 0.11
         \l_texnegar_fnt_titr_tl         { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
         \l_texnegar_fnt_titre_tl        { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
         \l_texnegar_fnt_traffic_tl      { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
         \l_texnegar_fnt_vahid_tl        { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
         \l_texnegar_fnt_vosta_tl        { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
         \l_texnegar_fnt_yaghut_tl       { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
         \l_texnegar_fnt_yagut_tl        { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
         \l_texnegar_fnt_yas_tl          { \tl_set:Nn \l_texnegar_skip_default_tl { 0.12
         \l_texnegar_fnt_yekan_tl        { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
         \l_texnegar_fnt_yermook_tl      { \tl_set:Nn \l_texnegar_skip_default_tl { 0.13
         \l_texnegar_fnt_zar_tl          { \tl_set:Nn \l_texnegar_skip_default_tl { 0.11
         \l_texnegar_fnt_ziba_tl         { \tl_set:Nn \l_texnegar_skip_default_tl { 0.11
         \l_texnegar_fnt_default_tl      { \tl_set:Nn \l_texnegar_skip_default_tl { 0.14
         \l_texnegar_fnt_noskip_tl       { \tl_set:Nn \l_texnegar_skip_default_tl { 0
      } { } { \tl_set:Nn \l_texnegar_skip_default_tl { #1 } }
  } ,
kashidastretch .default:n = \tl_set:Nn \l_texnegar_skip_default_tl { 0 em plus 0.5 em }

ligatures .code:n =
  {
    \tl_set:Nn \l_tmpa_tl { #1 }
    \tl_case:NnTF \l_tmpa_tl
      {
        \l_texnegar_lig_aalt_tl    { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
        \l_texnegar_lig_ccmp_tl    { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
        \l_texnegar_lig_dlig_tl    { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
        \l_texnegar_lig_fina_tl    { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
        \l_texnegar_lig_init_tl    { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
        \l_texnegar_lig_locl_tl    { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
        \l_texnegar_lig_medi_tl    { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
        \l_texnegar_lig_rlig_tl    { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
        \l_texnegar_lig_default_tl { \tl_set:Nx \l_texnegar_active_ligs_tl { \l_texnegar
      } { } { \tl_set:Nn \l_texnegar_active_ligs_tl { #1 } }
    \bool_set_true:N \l_texnegar_ligature_bool
  } ,
ligatures .default:n = \tl_set:Nn \l_texnegar_active_ligs_tl { \l_texnegar_lig_default_t
```

```
  color .code:n =
    {
      \tl_set:Nn \l_tmpa_tl { #1 }
      \tl_if_empty:NTF \l_tmpa_tl
        {
          \tl_set:Nx \l_texnegar_color_tl { \l_texnegar_col_default_tl }
        }
        {
          \tl_set:Nx \l_texnegar_color_tl { \l_tmpa_tl }
        }
      \bool_set_true:N  \l_texnegar_color_bool
      \sys_if_engine_luatex:T
        {
          \convertcolorspec{named}{\l_texnegar_color_tl}{rgb}\l_texnegar_color_rgb_tl
          \sys_if_engine_luatex:T
            {
              \directlua{l_texnegar_color_rgb_tl = "\l_texnegar_color_rgb_tl"}
            }
        }
    } ,

  hboxrecursion .code:n =
    {
      \tl_set:Nn \l_tmpa_tl { #1 }
      \tl_case:NnTF \l_tmpa_tl
        {
          \l_texnegar_hboxrecursion_off_tl
            {
              \bool_set_false:N \l_texnegar_hboxrecursion_bool
            }
          \l_texnegar_hboxrecursion_on_tl
            {
              \bool_set_true:N \l_texnegar_hboxrecursion_bool
            }
        } { } { \bool_set_false:N \l_texnegar_hboxrecursion_bool }
    } ,
  hboxrecursion .default:n = \bool_set_true:N \l_texnegar_hboxrecursion_bool ,

  vboxrecursion .code:n =
    {
      \tl_set:Nn \l_tmpa_tl { #1 }
      \tl_case:NnTF \l_tmpa_tl
        {
          \l_texnegar_vboxrecursion_off_tl
            {
              \bool_set_false:N \l_texnegar_vboxrecursion_bool
            }
          \l_texnegar_vboxrecursion_on_tl
            {
              \bool_set_true:N \l_texnegar_vboxrecursion_bool
            }
        } { } { \bool_set_false:N \l_texnegar_vboxrecursion_bool }
    } ,
  vboxrecursion .default:n = \bool_set_true:N \l_texnegar_vboxrecursion_bool ,
  }

\ProcessKeysOptions { texnegar }

\sys_if_engine_luatex:T
  {
    \NewDocumentCommand \KashidaHMFixOff {} { \directlua{StopStretching()} }
    \NewDocumentCommand \KashidaHMFixOn  {} { \directlua{StartStretching()} }
  }

\sys_if_engine_xetex:T
  {
    \NewDocumentCommand \KashidaHMFixOn {} { \bool_set_true:N \l_texnegar_kashida_fix_bool }
    \NewDocumentCommand \KashidaHMFixOff {} { \bool_set_false:N \l_texnegar_kashida_fix_bool
  }

\tex_let:D \KashidaOn \KashidaHMFixOn
\tex_let:D \KashidaOff \KashidaHMFixOff

\bool_if:NTF \l_texnegar_kashida_fix_bool
  {
    \tl_if_empty:NT \l_texnegar_skip_default_tl { \tl_set:Nn \l_texnegar_skip_default_tl  {
  }
  {
    \tl_set:NV \l_texnegar_skip_default_tl  \c_texnegar_skip_a_tl
  }

%% % \makeatletter
%% % \newif\if@Kashida@on
%% Becuase Vafa Khalighi has copied the above code (injecting the character uni+200E) in xep
23.0
%% (https://tug.org/svn/texlive/trunk/Master/texmf-dist/tex/xelatex/xepersian/kashida-
xepersian.def?revision=55165&view=co),
%% the following line of code is not needed in xepersian anymore.
%% % \newif\if@Kashida@XB@fix
%% % \makeatother

\bool_if:NF \l_texnegar_minimal_bool
  {
    \directlua{dofile(kpse.find_file("luatex-tools.lua"))}
    \input texnegar-luabidi.tex
  }

 \endinput
⟨/texnegar-ini-tex⟩
```

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

```
⟨*texnegar-common-kashida-tex⟩
\ProvidesExplFile {texnegar-common-kashida.tex} {2021-02-09} {0.1e} { Full implementation of

\bool_if:NT \l_texnegar_ligature_bool
{
  \clist_new:N \l_texnegar_ligatures_clist
  \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_tmpb_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_tmpb_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_tmpb_tl { } { }
      \tl_const:cx { \tl_use:N \l_tmpb_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
        }{
           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 } { O{\l_texnegar_line_break_penalty_int} O
    {
      \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 }
   }

  \cs_new_protected:Nn \texnegar_put_line_breaks:n
    {
      \tl_set:Nn \l_texnegar_line_break_tl { #1 }
      \regex_replace_all:nnN { ([])+ } { \ \0 \  \c{texnegar_line_break:}\  } \l_texnegar_li
      \tl_use:N \l_texnegar_line_break_tl
    }
}

 \endinput
⟨/texnegar-common-kashida-tex⟩
```

1.6 File: texnegar-xetex-kashida.tex

```
⟨*texnegar-xetex-kashida-tex⟩
\ProvidesExplFile {texnegar-xetex-kashida.tex} {2021-02-09} {0.1e} { Full implementation of

\newXeTeXintercharclass \c_texnegar_d_charclass % dual-joiner class
\newXeTeXintercharclass \c_texnegar_l_charclass % lam
\newXeTeXintercharclass \c_texnegar_r_charclass % right-joiner
\newXeTeXintercharclass \c_texnegar_a_charclass % alef
\newXeTeXintercharclass \c_texnegar_y_charclass % yeh

\tex_input:D { texnegar-common-kashida.tex }

\tl_set:Nn \l_texnegar_use_color_tl
  {
    \bool_if:NTF \l_texnegar_color_bool
      {
        \colorlet{default}{\l_texnegar_color_tl}
      }
      {
        \colorlet{default}{.}
      }
      \color{default}
  }

%% Partly adapted from the code provided by David Carlisle in:
%% https://tex.stackexchange.com/questions/356709/how-to-know-the-width-and-fill-the-glue-space-between-two-characters-when-using/356721#356721
\cs_new:Npn \texnegar_kashida_glyph #1
{
  \bool_if:NT \l_texnegar_kashida_fix_bool
  {
    \c_texnegar_lrm_int\tex_penalty:D 10000
    \mode_leave_vertical:
    \tex_global:D \tex_advance:D \l_texnegar_counter_int \c_one_int

    \tl_set:Nx \l_texnegar_pos_tl { p\tex_romannumeral:D \l_texnegar_counter_int }
    \tl_set:Nx \l_texnegar_zref_tl { z\tex_romannumeral:D \l_texnegar_counter_int }

    \zsaveposx{x_i_\l_texnegar_zref_tl}
    \tl_set:Nx \l_tmpa_tl
      {
        \iow_now:cx { @auxout }
        {
          \token_to_str:N \gdef \exp_after:wN \token_to_str:N \cs:w xi\l_texnegar_pos_tl \cs
        }
      }
    \l_tmpa_tl
    \skip_horizontal:n { #1 }
    \zsaveposx{x_f_\l_texnegar_zref_tl}
    \tl_set:Nx \l_tmpa_tl
      {
        \iow_now:cx { @auxout }
        {
          \token_to_str:N \gdef \exp_after:wN \token_to_str:N \cs:w xf\l_texnegar_pos_tl \cs
        }
      }
    \l_tmpa_tl
    \exp_after:wN
    \if_meaning:w
      \cs:w xi\l_texnegar_pos_tl \cs_end: \tex_relax:D
    \else:
      \dim_set:Nn \l_texnegar_diff_pos_dim
        {
          \dim_eval:n { \cs:w xi\l_texnegar_pos_tl \cs_end: sp - \cs:w xf\l_texnegar_pos_tl
        }
      \dim_compare:nTF { \l_texnegar_diff_pos_dim == 0sp }
        { }
        { \llap { \resizebox { \l_texnegar_diff_pos_dim \tex_relax:D } { \height } { \l_texn
    \fi:
  }
}

\cs_new:Npn \texnegar_kashida_leaders #1
{
  \bool_if:NT \l_texnegar_kashida_fix_bool
    {
      \tl_if_eq:NNTF \l_texnegar_gap_filler_tl  \l_texnegar_stretch_leaders_glyph_tl
        {
          \tl_set:Nx \l_texnegar_font_full_tl { \tex_fontname:D \tex_the:D \tex_font:D }
          \tl_set:Nx \l_texnegar_font_name_tl { \l_texnegar_font_full_tl }
          \tl_set:Nx \l_texnegar_font_init_tl { \l_texnegar_font_name_tl }
          \regex_replace_once:nnN { ^"\[?(HM)[\_\ ](X|F).* } { \1\2 } \l_texnegar_font_init_
          \tl_set:Nn \l_tmpa_tl { HMF }
          \tl_set:Nn \l_tmpb_tl { HMX }
          \bool_if:nTF { \str_if_eq_p:NN { \l_texnegar_font_init_tl } { \l_tmpa_tl } || \str
            {
              \hbox_set:Nn \l_texnegar_ksh_box { \l_texnegar_use_color_tl \XeTeXglyph\XeTeXg
              \c_texnegar_zwj_int \tex_penalty:D 10000
              \tex_leaders:D \copy\l_texnegar_ksh_box \skip_horizontal:n { #1 }
              \c_texnegar_zwj_int
            }
            {
              \msg_error:nnx { texnegar } { hm-series-font-not-found } { \l_texnegar_font_na
            }
        }
        {
          %% Partly adapted from the code provided by Jonathan Kew in:
          %% https://tug.org/pipermail/xetex/2009-February/012307.html.
          %% Somebody notified me that the code in 'kashida-xepersian.def' from xepersian
          %% package is an exact copy of Jonathan Kew's code. Being unaware of this, in
          %% the earlier versions of this package I made a mistake and acknowledged
          %% Vafa Khalighi instead of Jonathan Kew. A sincere thank you to Jonathan Kew
          %% for his excellent code.
          \c_texnegar_lrm_int\c_texnegar_zwj_int
          {\l_texnegar_use_color_tl\tex_penalty:D 10000
          \tex_leaders:D \tex_hrule:D height \XeTeXglyphbounds \c_texnegar_two_int
          \int_use:N \XeTeXcharglyph \c_texnegar_ksh_int depth \XeTeXglyphbounds \c_texnegar
          \int_use:N \XeTeXcharglyph \c_texnegar_ksh_int \skip_horizontal:n { #1 }
          }
          \c_texnegar_zwj_int
        }
    }
}

\XeTeXinterchartokenstate = 1

\clist_set:Nn \l_texnegar_a_clist { 0622,0623,0625,0627 } %
\clist_map_inline:Nn \l_texnegar_a_clist
  {
    \XeTeXcharclass "#1 \c_texnegar_a_charclass
  }

\clist_set:Nn \l_texnegar_d_clist { 0626,0628,062A,062B,062C,062D,062E,0633,0634,0635,0636,0
\clist_map_inline:Nn \l_texnegar_d_clist
  {
    \XeTeXcharclass "#1 \c_texnegar_d_charclass
  }

\clist_set:Nn \l_texnegar_l_clist { 0644 } %
\clist_map_inline:Nn \l_texnegar_l_clist
  {
    \XeTeXcharclass "#1 \c_texnegar_l_charclass
  }

\clist_set:Nn \l_texnegar_r_clist { 0624,0629,062F,0630,0631,0632,0648,0698 } % ,,,,,,,
\clist_map_inline:Nn \l_texnegar_r_clist
  {
    \XeTeXcharclass "#1 \c_texnegar_r_charclass
  }

\clist_set:Nn \l_texnegar_y_clist { 0649,064A,06CC } % ,,
\clist_map_inline:Nn \l_texnegar_y_clist
  {
    \XeTeXcharclass "#1 \c_texnegar_y_charclass
  }

\tl_if_eq:NNTF  \l_texnegar_gap_filler_tl  \l_texnegar_stretch_glyph_tl {
  \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_y_charclass = {
    \bool_if:NTF \l_texnegar_kashida_fix_bool
    { \c_texnegar_zwj_int \texnegar_kashida_glyph \l_texnegar_skip_default_tl \c_texnegar_zw
    { \c_texnegar_zwj_int \texnegar_kashida_glyph \c_texnegar_skip_a_tl \c_texnegar_zwj_int
  }
  \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_y_charclass = {
    \bool_if:NTF \l_texnegar_kashida_fix_bool
    { \c_texnegar_zwj_int \texnegar_kashida_glyph \l_texnegar_skip_default_tl \c_texnegar_zw
    { \c_texnegar_zwj_int \texnegar_kashida_glyph \c_texnegar_skip_a_tl \c_texnegar_zwj_int
  }
  \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_d_charclass = { \c_texnegar_zwj_in
  \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_d_charclass = { \c_texnegar_zwj_in
  \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_d_charclass = { \c_texnegar_zwj_in
  \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_l_charclass = { \c_texnegar_zwj_in
  \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_l_charclass = { \c_texnegar_zwj_in
  \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_r_charclass = { \c_texnegar_zwj_in
  \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_a_charclass = { \c_texnegar_zwj_in
  \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_r_charclass = { \c_texnegar_zwj_in
  \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_a_charclass = { }
}
{
  \bool_if:nTF {
    \tl_if_eq_p:NN  \l_texnegar_gap_filler_tl  \l_texnegar_stretch_leaders_glyph_tl ||
    \tl_if_eq_p:NN  \l_texnegar_gap_filler_tl  \l_texnegar_stretch_leaders_hrule_tl
  }
  {
    \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_y_charclass = {
      \bool_if:NTF \l_texnegar_kashida_fix_bool
      { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
      { \texnegar_kashida_leaders \c_texnegar_skip_a_tl }
    }
    \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_y_charclass = {
      \bool_if:NTF \l_texnegar_kashida_fix_bool
      { \texnegar_kashida_leaders \l_texnegar_skip_default_tl }
      { \texnegar_kashida_leaders \c_texnegar_skip_a_tl }
    }
    \XeTeXinterchartoks \c_texnegar_y_charclass \c_texnegar_d_charclass = { \texnegar_kashid
    \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_d_charclass = { \texnegar_kashid
    \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_d_charclass = { \texnegar_kashid
    \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_l_charclass = { \texnegar_kashid
    \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_l_charclass = { \texnegar_kashid
    \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_r_charclass = { \texnegar_kashid
    \XeTeXinterchartoks \c_texnegar_d_charclass \c_texnegar_a_charclass = { \texnegar_kashid
    \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_r_charclass = { \texnegar_kashid
    \XeTeXinterchartoks \c_texnegar_l_charclass \c_texnegar_a_charclass = { }
  }
  {
    \msg_error:nnx { texnegar } { error-value-not-available-for-kashida-option } { \l_texneg
  }
}

 \endinput
⟨/texnegar-xetex-kashida-tex⟩
```

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

```
⟨*texnegar-char-table-lua⟩
--
```

```
-- This is file ‘texnegar-char-table.lua’,
-- generated with the docstrip utility.
--
-- The original source files were:
--
-- texnegar.dtx  (with options: ‘texnegar-char-table-lua’)
--
-- Copyright (C) 2020-2021 Hossein Movahhedian
--
-- It may be distributed and/or modified under the LaTeX Project Public License,
-- version 1.3c or higher (your choice). The latest version of
-- this license is at: http://www.latex-project.org/lppl.txt
--
-- texnegar_char_table        = texnegar_char_table or {}
-- local texnegar_char_table  = texnegar_char_table
-- texnegar_char_table.module = {
--     name                   = "texnegar_char_table",
--     version                = "0.1e",
--     date                   = "2021-02-09",
--     description            = "Full implementation of kashida feature in XeLaTex and LuaLa
--     author                 = "Hossein Movahhedian",
--     copyright              = "Hossein Movahhedian",
--     license                = "LPPL v1.3c"
-- }
--
-- -- ^^A%%  texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
-- local err, warn, info, log = luatexbase.provides_module(texnegar_char_table.module)
-- texnegar_char_table.log     = log  or (function (s) luatexbase.module_info("texnegar_char
-- texnegar_char_table.warning = warn or (function (s) luatexbase.module_warning("texnegar_c
-- texnegar_char_table.error   = err  or (function (s) luatexbase.module_error("texnegar_cha

local peCharTableDiacritic  = {
  [1611]  = utf8.char(1611),   -- "",  utf8.codepoint("") == 1611,   "\u{064B}", ARABIC-FATHATAN
  [1612]  = utf8.char(1612),   -- "",  utf8.codepoint("") == 1612,   "\u{064C}", ARABIC-DAMMATAN
  [1613]  = utf8.char(1613),   -- "",  utf8.codepoint("") == 1613,   "\u{064D}", ARABIC-KASRATAN
  [1614]  = utf8.char(1614),   -- "",  utf8.codepoint("") == 1614,   "\u{064E}", ARABIC-FATHA
  [1615]  = utf8.char(1615),   -- "",  utf8.codepoint("") == 1615,   "\u{064F}", ARABIC-DAMMA
  [1616]  = utf8.char(1616),   -- "",  utf8.codepoint("") == 1616,   "\u{0650}", ARABIC-KASRA
  [1617]  = utf8.char(1617),   -- "",  utf8.codepoint("") == 1617,   "\u{0651}", ARABIC-SHADDA
  [1618]  = utf8.char(1618),   -- "",  utf8.codepoint("") == 1618,   "\u{0652}", ARABIC-SUKUN
  [1619]  = utf8.char(1619),   -- "",  utf8.codepoint("") == 1619,   "\u{0653}", ARABIC-MADDA ABOVE
  [1620]  = utf8.char(1620),   -- "",  utf8.codepoint("") == 1620,   "\u{0654}", ARABIC-HAMZA ABOVE
  [1621]  = utf8.char(1621),   -- "",  utf8.codepoint("") == 1621,   "\u{0655}", ARABIC-HAMZA BELOW
  [1622]  = utf8.char(1622),   -- "",  utf8.codepoint("") == 1622,   "\u{0656}", ARABIC-
SUBSCRIPT ALEF
  [1623]  = utf8.char(1623),   -- "",  utf8.codepoint("") == 1623,   "\u{0657}", ARABIC-
INVERTED DAMMA
  [1624]  = utf8.char(1624),   -- "",  utf8.codepoint("") == 1624,   "\u{0658}", ARABIC-
MARK NOON GHUNNA
  [1625]  = utf8.char(1625),   -- "",  utf8.codepoint("") == 1625,   "\u{0659}", ARABIC-
ZWARAKAY
  [1648]  = utf8.char(1648),   -- "",  utf8.codepoint("") == 1648,   "\u{0670}", ARABIC-
SUPERSCRIPT ALEF
  [64606] = utf8.char(64606),  -- "",  utf8.codepoint("") == 64606,  "\u{FC5E}", ARABIC-
LIGATURE SHADDA WITH DAMMATAN ISOLATED FORM
  [64607] = utf8.char(64607),  -- "",  utf8.codepoint("") == 64607,  "\u{FC5F}", ARABIC-
LIGATURE SHADDA WITH KASRATAN ISOLATED FORM
  [64608] = utf8.char(64608),  -- "",  utf8.codepoint("") == 64608,  "\u{FC60}", ARABIC-
LIGATURE SHADDA WITH FATHA ISOLATED FORM
  [64609] = utf8.char(64609),  -- "",  utf8.codepoint("") == 64609,  "\u{FC61}", ARABIC-
LIGATURE SHADDA WITH DAMMA ISOLATED FORM
  [64610] = utf8.char(64610),  -- "",  utf8.codepoint("") == 64610,  "\u{FC62}", ARABIC-
LIGATURE SHADDA WITH KASRA ISOLATED FORM
  [64611] = utf8.char(64611),  -- "",  utf8.codepoint("") == 64611,  "\u{FC63}", ARABIC-
LIGATURE SHADDA WITH SUPERSCRIPT ALEF ISOLATED FORM
}

local peCharTableDigit  = {
  [1632] = utf8.char(1632),  -- "",  utf8.codepoint("") == 1632,  "\u{0660}", ARABIC-
INDIC DIGIT ZERO
  [1633] = utf8.char(1633),  -- "",  utf8.codepoint("") == 1633,  "\u{0661}", ARABIC-
INDIC DIGIT ONE
  [1634] = utf8.char(1634),  -- "",  utf8.codepoint("") == 1634,  "\u{0662}", ARABIC-
INDIC DIGIT TWO
  [1635] = utf8.char(1635),  -- "",  utf8.codepoint("") == 1635,  "\u{0663}", ARABIC-
INDIC DIGIT THREE
  [1636] = utf8.char(1636),  -- "",  utf8.codepoint("") == 1636,  "\u{0664}", ARABIC-
INDIC DIGIT FOUR
  [1637] = utf8.char(1637),  -- "",  utf8.codepoint("") == 1637,  "\u{0665}", ARABIC-
INDIC DIGIT FIVE
  [1638] = utf8.char(1638),  -- "",  utf8.codepoint("") == 1638,  "\u{0666}", ARABIC-
INDIC DIGIT SIX
  [1639] = utf8.char(1639),  -- "",  utf8.codepoint("") == 1639,  "\u{0667}", ARABIC-
INDIC DIGIT SEVEN
  [1640] = utf8.char(1640),  -- "",  utf8.codepoint("") == 1640,  "\u{0668}", ARABIC-
INDIC DIGIT EIGHT
  [1641] = utf8.char(1641),  -- "",  utf8.codepoint("") == 1641,  "\u{0669}", ARABIC-
INDIC DIGIT NINE
  [1780] = utf8.char(1780),  -- "",  utf8.codepoint("") == 1780,  "\u{06F4}", EXTENDED ARABI
INDIC DIGIT FOUR
  [1781] = utf8.char(1781),  -- "",  utf8.codepoint("") == 1781,  "\u{06F5}", EXTENDED ARABI
INDIC DIGIT FIVE
  [1782] = utf8.char(1782),  -- "",  utf8.codepoint("") == 1782,  "\u{06F6}", EXTENDED ARABI
INDIC DIGIT SIX
}

local peCharTablePunctuation  = {
  [1548] = utf8.char(1548),  -- "",  utf8.codepoint("") == 1548,  "\u{060C}", ARABIC COMMA
  [1549] = utf8.char(1549),  -- "",  utf8.codepoint("") == 1549,  "\u{060D}", ARABIC DATE SE
  [1563] = utf8.char(1563),  -- "",  utf8.codepoint("") == 1563,  "\u{061B}", ARABIC SEMICOL
  [1567] = utf8.char(1567),  -- "",  utf8.codepoint("") == 1567,  "\u{061F}", ARABIC QUESTIO
  [1642] = utf8.char(1642),  -- "",  utf8.codepoint("") == 1642,  "\u{066A}", ARABIC PERCENT
  [1643] = utf8.char(1643),  -- "",  utf8.codepoint("") == 1643,  "\u{066B}", ARABIC DECIMAL
  [1644] = utf8.char(1644),  -- "",  utf8.codepoint("") == 1644,  "\u{066C}", ARABIC THOUSAN
  [1645] = utf8.char(1645),  -- "",  utf8.codepoint("") == 1645,  "\u{066D}", ARABIC FIVE PO
}

local peCharTable  = {
  [1569] = utf8.char(1569),    -- "",  utf8.codepoint("") == 1569,  "\u{0621}", ARABIC LETTE
  [1570] = utf8.char(1570),    -- "",  utf8.codepoint("") == 1570,  "\u{0622}", ARABIC LETTE
  [1571] = utf8.char(1571),    -- "",  utf8.codepoint("") == 1571,  "\u{0623}", ARABIC LETTE
  [1572] = utf8.char(1572),    -- "",  utf8.codepoint("") == 1572,  "\u{0624}", ARABIC LETTE
  [1573] = utf8.char(1573),    -- "",  utf8.codepoint("") == 1573,  "\u{0625}", ARABIC LETTE
  [1574] = utf8.char(1574),    -- "",  utf8.codepoint("") == 1574,  "\u{0626}", ARABIC LETTE
  [1575] = utf8.char(1575),    -- "",  utf8.codepoint("") == 1575,  "\u{0627}", ARABIC LETTE
  [1576] = utf8.char(1576),    -- "",  utf8.codepoint("") == 1576,  "\u{0628}", ARABIC LETTE
  [1577] = utf8.char(1577),    -- "",  utf8.codepoint("") == 1577,  "\u{0629}", ARABIC LETTE
  [1578] = utf8.char(1578),    -- "",  utf8.codepoint("") == 1578,  "\u{062A}", ARABIC LETTE
  [1579] = utf8.char(1579),    -- "",  utf8.codepoint("") == 1579,  "\u{062B}", ARABIC LETTE
  [1580] = utf8.char(1580),    -- "",  utf8.codepoint("") == 1580,  "\u{062C}", ARABIC LETTE
  [1581] = utf8.char(1581),    -- "",  utf8.codepoint("") == 1581,  "\u{062D}", ARABIC LETTE
  [1582] = utf8.char(1582),    -- "",  utf8.codepoint("") == 1582,  "\u{062E}", ARABIC LETTE
  [1583] = utf8.char(1583),    -- "",  utf8.codepoint("") == 1583,  "\u{062F}", ARABIC LETTE
  [1584] = utf8.char(1584),    -- "",  utf8.codepoint("") == 1584,  "\u{0630}", ARABIC LETTE
  [1585] = utf8.char(1585),    -- "",  utf8.codepoint("") == 1585,  "\u{0631}", ARABIC LETTE
  [1586] = utf8.char(1586),    -- "",  utf8.codepoint("") == 1586,  "\u{0632}", ARABIC LETTE
  [1587] = utf8.char(1587),    -- "",  utf8.codepoint("") == 1587,  "\u{0633}", ARABIC LETTE
  [1588] = utf8.char(1588),    -- "",  utf8.codepoint("") == 1588,  "\u{0634}", ARABIC LETTE
  [1589] = utf8.char(1589),    -- "",  utf8.codepoint("") == 1589,  "\u{0635}", ARABIC LETTE
  [1590] = utf8.char(1590),    -- "",  utf8.codepoint("") == 1590,  "\u{0636}", ARABIC LETTE
  [1591] = utf8.char(1591),    -- "",  utf8.codepoint("") == 1591,  "\u{0637}", ARABIC LETTE
  [1592] = utf8.char(1592),    -- "",  utf8.codepoint("") == 1592,  "\u{0638}", ARABIC LETTE
  [1593] = utf8.char(1593),    -- "",  utf8.codepoint("") == 1593,  "\u{0639}", ARABIC LETTE
  [1594] = utf8.char(1594),    -- "",  utf8.codepoint("") == 1594,  "\u{063A}", ARABIC LETTE
  [1601] = utf8.char(1601),    -- "",  utf8.codepoint("") == 1601,  "\u{0641}", ARABIC LETTE
  [1602] = utf8.char(1602),    -- "",  utf8.codepoint("") == 1602,  "\u{0642}", ARABIC LETTE
  [1603] = utf8.char(1603),    -- "",  utf8.codepoint("") == 1603,  "\u{0643}", ARABIC LETTE
  [1604] = utf8.char(1604),    -- "",  utf8.codepoint("") == 1604,  "\u{0644}", ARABIC LETTE
  [1605] = utf8.char(1605),    -- "",  utf8.codepoint("") == 1605,  "\u{0645}", ARABIC LETTE
  [1606] = utf8.char(1606),    -- "",  utf8.codepoint("") == 1606,  "\u{0646}", ARABIC LETTE
  [1607] = utf8.char(1607),    -- "",  utf8.codepoint("") == 1607,  "\u{0647}", ARABIC LETTE
  [1608] = utf8.char(1608),    -- "",  utf8.codepoint("") == 1608,  "\u{0648}", ARABIC LETTE
  [1609] = utf8.char(1609),    -- "",  utf8.codepoint("") == 1609,  "\u{0649}", ARABIC LETTE
  [1610] = utf8.char(1610),    -- "",  utf8.codepoint("") == 1610,  "\u{064A}", ARABIC LETTE
  [1662] = utf8.char(1662),    -- "",  utf8.codepoint("") == 1662,  "\u{067E}", ARABIC LETTE
  [1670] = utf8.char(1670),    -- "",  utf8.codepoint("") == 1670,  "\u{0686}", ARABIC LETTE
  [1688] = utf8.char(1688),    -- "",  utf8.codepoint("") == 1688,  "\u{0698}", ARABIC LETTE
  [1705] = utf8.char(1705),    -- "",  utf8.codepoint("") == 1705,  "\u{06A9}", ARABIC LETTE
  [1706] = utf8.char(1706),    -- "",  utf8.codepoint("") == 1706,  "\u{06AA}", ARABIC LETTE
  [1711] = utf8.char(1711),    -- "",  utf8.codepoint("") == 1711,  "\u{06AF}", ARABIC LETTE
  [1726] = utf8.char(1726),    -- "",  utf8.codepoint("") == 1726,  "\u{06BE}", ARABIC LETTE
  [1728] = utf8.char(1728),    -- "",  utf8.codepoint("") == 1728,  "\u{06C0}", ARABIC LETTE
  [1740] = utf8.char(1740),    -- "",  utf8.codepoint("") == 1740,  "\u{06CC}", ARABIC LETTE
  [1749] = utf8.char(1749),    -- "",  utf8.codepoint("") == 1740,  "\u{06D5}", ARABIC LETTE
  [65275] = utf8.char(65275),  -- "",  utf8.codepoint("") == 65275,  "\u{FEFB}", ARABIC LIGA
  [65276] = utf8.char(65276),  -- "",  utf8.codepoint("") == 65276,  "\u{FEFC}", ARABIC LIGA
}

local peCharTableInitial  = {
  [64344] = utf8.char(64344),  -- "",  utf8.codepoint("") == 64344,  "\u{FB58}", INITIAL FOR
  [64380] = utf8.char(64380),  -- "",  utf8.codepoint("") == 64380,  "\u{FB7C}", INITIAL FOR
  [64400] = utf8.char(64400),  -- "",  utf8.codepoint("") == 64400,  "\u{FB90}", INITIAL FOR
  [64404] = utf8.char(64404),  -- "",  utf8.codepoint("") == 64404,  "\u{FB94}", INITIAL FOR
  [64510] = utf8.char(64510),  -- "",  utf8.codepoint("") == 64510,  "\u{FBFE}", INITIAL FOR
  [65169] = utf8.char(65169),  -- "",  utf8.codepoint("") == 65169,  "\u{FE91}", INITIAL FOR
  [65175] = utf8.char(65175),  -- "",  utf8.codepoint("") == 65175,  "\u{FE97}", INITIAL FOR
  [65179] = utf8.char(65179),  -- "",  utf8.codepoint("") == 65179,  "\u{FE9B}", INITIAL FOR
  [65183] = utf8.char(65183),  -- "",  utf8.codepoint("") == 65183,  "\u{FE9F}", INITIAL FOR
  [65187] = utf8.char(65187),  -- "",  utf8.codepoint("") == 65187,  "\u{FEA3}", INITIAL FOR
  [65191] = utf8.char(65191),  -- "",  utf8.codepoint("") == 65191,  "\u{FEA7}", INITIAL FOR
  [65203] = utf8.char(65203),  -- "",  utf8.codepoint("") == 65203,  "\u{FEB3}", INITIAL FOR
  [65207] = utf8.char(65207),  -- "",  utf8.codepoint("") == 65207,  "\u{FEB7}", INITIAL FOR
  [65211] = utf8.char(65211),  -- "",  utf8.codepoint("") == 65211,  "\u{FEBB}", INITIAL FOR
  [65215] = utf8.char(65215),  -- "",  utf8.codepoint("") == 65215,  "\u{FEBF}", INITIAL FOR
  [65219] = utf8.char(65219),  -- "",  utf8.codepoint("") == 65219,  "\u{FEC3}", INITIAL FOR
  [65223] = utf8.char(65223),  -- "",  utf8.codepoint("") == 65223,  "\u{FEC7}", INITIAL FOR
  [65227] = utf8.char(65227),  -- "",  utf8.codepoint("") == 65227,  "\u{FECB}", INITIAL FOR
  [65231] = utf8.char(65231),  -- "",  utf8.codepoint("") == 65231,  "\u{FECF}", INITIAL FOR
  [65235] = utf8.char(65235),  -- "",  utf8.codepoint("") == 65235,  "\u{FED3}", INITIAL FOR
  [65239] = utf8.char(65239),  -- "",  utf8.codepoint("") == 65239,  "\u{FED7}", INITIAL FOR
  [65243] = utf8.char(65243),  -- "",  utf8.codepoint("") == 65243,  "\u{FEDB}", INITIAL FOR
  [65247] = utf8.char(65247),  -- "",  utf8.codepoint("") == 65247,  "\u{FEDF}", INITIAL FOR
  [65251] = utf8.char(65251),  -- "",  utf8.codepoint("") == 65251,  "\u{FEE3}", INITIAL FOR
  [65255] = utf8.char(65255),  -- "",  utf8.codepoint("") == 65255,  "\u{FEE7}", INITIAL FOR
  [65259] = utf8.char(65259),  -- "",  utf8.codepoint("") == 65259,  "\u{FEEB}", INITIAL FOR
  [65267] = utf8.char(65267),  -- "",  utf8.codepoint("") == 65267,  "\u{FEF3}", INITIAL FOR
}

local peCharTableMedial  = {
  [1600]  = utf8.char(1600),   -- "",  utf8.codepoint("") == 1600,   "\u{0640}", ARABIC TATW
  [64345] = utf8.char(64345),  -- "",  utf8.codepoint("") == 64345,  "\u{FB59}", MEDIAL FORM
  [64381] = utf8.char(64381),  -- "",  utf8.codepoint("") == 64381,  "\u{FB7D}", MEDIAL FORM
  [64401] = utf8.char(64401),  -- "",  utf8.codepoint("") == 64401,  "\u{FB91}", MEDIAL FORM
  [64405] = utf8.char(64405),  -- "",  utf8.codepoint("") == 64405,  "\u{FB95}", MEDIAL FORM
  [64425] = utf8.char(64425),  -- "",  utf8.codepoint("") == 64425,  "\u{FBA9}", MEDIAL FORM
  [64429] = utf8.char(64429),  -- "",  utf8.codepoint("") == 64429,  "\u{FBAD}", MEDIAL FORM
  [64511] = utf8.char(64511),  -- "",  utf8.codepoint("") == 64511,  "\u{FBFF}", MEDIAL FORM
  [65170] = utf8.char(65170),  -- "",  utf8.codepoint("") == 65170,  "\u{FE92}", MEDIAL FORM
  [65176] = utf8.char(65176),  -- "",  utf8.codepoint("") == 65176,  "\u{FE98}", MEDIAL FORM
  [65180] = utf8.char(65180),  -- "",  utf8.codepoint("") == 65180,  "\u{FE9C}", MEDIAL FORM
  [65184] = utf8.char(65184),  -- "",  utf8.codepoint("") == 65184,  "\u{FEA0}", MEDIAL FORM
  [65188] = utf8.char(65188),  -- "",  utf8.codepoint("") == 65188,  "\u{FEA4}", MEDIAL FORM
  [65192] = utf8.char(65192),  -- "",  utf8.codepoint("") == 65192,  "\u{FEA8}", MEDIAL FORM
  [65204] = utf8.char(65204),  -- "",  utf8.codepoint("") == 65204,  "\u{FEB4}", MEDIAL FORM
  [65208] = utf8.char(65208),  -- "",  utf8.codepoint("") == 65208,  "\u{FEB8}", MEDIAL FORM
  [65212] = utf8.char(65212),  -- "",  utf8.codepoint("") == 65212,  "\u{FEBC}", MEDIAL FORM
  [65216] = utf8.char(65216),  -- "",  utf8.codepoint("") == 65216,  "\u{FEC0}", MEDIAL FORM
  [65220] = utf8.char(65220),  -- "",  utf8.codepoint("") == 65220,  "\u{FEC4}", MEDIAL FORM
  [65224] = utf8.char(65224),  -- "",  utf8.codepoint("") == 65224,  "\u{FEC8}", MEDIAL FORM
  [65228] = utf8.char(65228),  -- "",  utf8.codepoint("") == 65228,  "\u{FECC}", MEDIAL FORM
  [65232] = utf8.char(65232),  -- "",  utf8.codepoint("") == 65232,  "\u{FED0}", MEDIAL FORM
  [65236] = utf8.char(65236),  -- "",  utf8.codepoint("") == 65236,  "\u{FED4}", MEDIAL FORM
  [65240] = utf8.char(65240),  -- "",  utf8.codepoint("") == 65240,  "\u{FED8}", MEDIAL FORM
  [65244] = utf8.char(65244),  -- "",  utf8.codepoint("") == 65244,  "\u{FEDC}", MEDIAL FORM
  [65248] = utf8.char(65248),  -- "",  utf8.codepoint("") == 65248,  "\u{FEE0}", MEDIAL FORM
  [65252] = utf8.char(65252),  -- "",  utf8.codepoint("") == 65252,  "\u{FEE4}", MEDIAL FORM
  [65256] = utf8.char(65256),  -- "",  utf8.codepoint("") == 65256,  "\u{FEE8}", MEDIAL FORM
  [65260] = utf8.char(65260),  -- "",  utf8.codepoint("") == 65260,  "\u{FEEC}", MEDIAL FORM
  [65268] = utf8.char(65268),  -- "",  utf8.codepoint("") == 65268,  "\u{FEF4}", MEDIAL FORM
}

local peCharTableFinal  = {
  [64343] = utf8.char(64343),  -- "",  utf8.codepoint("") == 64343,  "\u{FB57}", FINAL FORM
  [64379] = utf8.char(64379),  -- "",  utf8.codepoint("") == 64379,  "\u{FB7B}", FINAL FORM
  [64395] = utf8.char(64395),  -- "",  utf8.codepoint("") == 64395,  "\u{FB8B}", FINAL FORM
  [64399] = utf8.char(64399),  -- "",  utf8.codepoint("") == 64399,  "\u{FB8F}", FINAL FORM
  [64403] = utf8.char(64403),  -- "",  utf8.codepoint("") == 64403,  "\u{FB93}", FINAL FORM
  [64421] = utf8.char(64421),  -- "",  utf8.codepoint("") == 64421,  "\u{FBA5}", FINAL FORM
  [64509] = utf8.char(64509),  -- "",  utf8.codepoint("") == 64509,  "\u{FBFD}", FINAL FORM
  [65166] = utf8.char(65166),  -- "",  utf8.codepoint("") == 65166,  "\u{FE8E}", FINAL FORM
  [65168] = utf8.char(65168),  -- "",  utf8.codepoint("") == 65168,  "\u{FE90}", FINAL FORM
  [65172] = utf8.char(65172),  -- "",  utf8.codepoint("") == 65172,  "\u{FE94}", FINAL FORM
  [65174] = utf8.char(65174),  -- "",  utf8.codepoint("") == 65174,  "\u{FE96}", FINAL FORM
  [65178] = utf8.char(65178),  -- "",  utf8.codepoint("") == 65178,  "\u{FE9A}", FINAL FORM
  [65182] = utf8.char(65182),  -- "",  utf8.codepoint("") == 65182,  "\u{FE9E}", FINAL FORM
  [65186] = utf8.char(65186),  -- "",  utf8.codepoint("") == 65186,  "\u{FEA2}", FINAL FORM
  [65190] = utf8.char(65190),  -- "",  utf8.codepoint("") == 65190,  "\u{FEA6}", FINAL FORM
  [65194] = utf8.char(65194),  -- "",  utf8.codepoint("") == 65194,  "\u{FEAA}", FINAL FORM
  [65196] = utf8.char(65196),  -- "",  utf8.codepoint("") == 65196,  "\u{FEAC}", FINAL FORM
  [65198] = utf8.char(65198),  -- "",  utf8.codepoint("") == 65198,  "\u{FEAE}", FINAL FORM
  [65200] = utf8.char(65200),  -- "",  utf8.codepoint("") == 65200,  "\u{FEB0}", FINAL FORM
  [65202] = utf8.char(65202),  -- "",  utf8.codepoint("") == 65202,  "\u{FEB2}", FINAL FORM
  [65206] = utf8.char(65206),  -- "",  utf8.codepoint("") == 65206,  "\u{FEB6}", FINAL FORM
  [65210] = utf8.char(65210),  -- "",  utf8.codepoint("") == 65210,  "\u{FEBA}", FINAL FORM
  [65214] = utf8.char(65214),  -- "",  utf8.codepoint("") == 65214,  "\u{FEBE}", FINAL FORM
  [65218] = utf8.char(65218),  -- "",  utf8.codepoint("") == 65218,  "\u{FEC2}", FINAL FORM
  [65222] = utf8.char(65222),  -- "",  utf8.codepoint("") == 65222,  "\u{FEC6}", FINAL FORM
  [65226] = utf8.char(65226),  -- "",  utf8.codepoint("") == 65226,  "\u{FECA}", FINAL FORM
  [65230] = utf8.char(65230),  -- "",  utf8.codepoint("") == 65230,  "\u{FECE}", FINAL FORM
  [65234] = utf8.char(65234),  -- "",  utf8.codepoint("") == 65234,  "\u{FED2}", FINAL FORM
  [65238] = utf8.char(65238),  -- "",  utf8.codepoint("") == 65238,  "\u{FED6}", FINAL FORM
  [65242] = utf8.char(65242),  -- "",  utf8.codepoint("") == 65242,  "\u{FEDA}", FINAL FORM
  [65246] = utf8.char(65246),  -- "",  utf8.codepoint("") == 65246,  "\u{FEDE}", FINAL FORM
  [65250] = utf8.char(65250),  -- "",  utf8.codepoint("") == 65250,  "\u{FEE2}", FINAL FORM
  [65254] = utf8.char(65254),  -- "",  utf8.codepoint("") == 65254,  "\u{FEE6}", FINAL FORM
  [65258] = utf8.char(65258),  -- "",  utf8.codepoint("") == 65258,  "\u{FEEA}", FINAL FORM
  [65262] = utf8.char(65262),  -- "",  utf8.codepoint("") == 65262,  "\u{FEEE}", FINAL FORM
  [65264] = utf8.char(65264),  -- "",  utf8.codepoint("") == 65264,  "\u{FEF0}", FINAL FORM
  [65266] = utf8.char(65266),  -- "",  utf8.codepoint("") == 65266,  "\u{FEF2}", FINAL FORM
  [65276] = utf8.char(65276),  -- "",  utf8.codepoint("") == 65276,  "\u{FEFC}", FINAL FORM
}

return peCharTableInitial, peCharTableMedial, peCharTableFinal, peCharTableDiacritic
--
--
-- End of file ‘texnegar-char-table.lua’.
⟨/texnegar-char-table-lua⟩
```

1.8 File: `texnegar.lua`

```
⟨*texnegar-lua⟩
--
-- This is file ‘texnegar.lua’,
-- generated with the docstrip utility.
--
-- The original source files were:
--
-- texnegar.dtx  (with options: ‘texnegar-lua’)
--
-- Copyright (C) 2020-2021 Hossein Movahhedian
--
-- It may be distributed and/or modified under the LaTeX Project Public License,
-- version 1.3c or higher (your choice). The latest version of
-- this license is at: http://www.latex-project.org/lppl.txt
--
-- texnegar          = texnegar or {}
-- local texnegar    = texnegar
-- texnegar.module   = {
--     name          = "texnegar",
--     version       = "0.1e",
--     date          = "2021-02-09",
--     description   = "Full implementation of kashida feature in XeLaTex and LuaLaTeX",
--     author        = "Hossein Movahhedian",
--     copyright     = "Hossein Movahhedian",
--     license       = "LPPL v1.3c"
-- }
--
-- -- ^^A%%  texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
-- local err, warn, info, log = luatexbase.provides_module(texnegar.module)
-- texnegar.log     = log  or (function (s) luatexbase.module_info("texnegar", s)    end)
-- texnegar.warning = warn or (function (s) luatexbase.module_warning("texnegar", s) end)
-- texnegar.error   = err  or (function (s) luatexbase.module_error("texnegar", s)   end)

local l_texnegar_kashida_fontfamily_bool = token.create("l_texnegar_kashida_fontfamily_bool"

local debug_getinfo = debug.getinfo
local string_format = string.format

function TableLength(t)
    local i = 0
    for _ in pairs(t) do
        i = i + 1
    end
    return i
```

```
end

tex.enableprimitives('',tex.extraprimitives ())

local range_tble = {
    [1536] = 1791,
    [1872] = 1919,
    [2208] = 2274,
    [8204] = 8297,
    [64336] = 65023,
    [65136] = 65279,
    [126464] = 126719,
    [983040] = 1048575
  }

local tbl_fonts_used = { }
local tbl_fonts_chars = { }
local tbl_fonts_chars_init = { }
local tbl_fonts_chars_medi = { }
local tbl_fonts_chars_fina = { }

local pattern_list = {
  ".*%.(ini)t?$",  ".*%.(ini)t?%..*",
  ".*%.(med)i?$",  ".*%.(med)i?%..*",
  ".*%.(fin)a?$",  ".*%.(fin)a?%..*",

  ".*_(ini)t?$",    ".*_(ini)t?_.*",
  ".*_(med)i?$",    ".*_(med)i?_.*",
  ".*_(fin)a?$",    ".*_(fin)a?_.*",
}

function GetFontsChars()
    local funcName    = debug_getinfo(1).name
    local funcNparams = debug_getinfo(1).nparams

    for f_num = 1, font.max() do
        local f_tmp = font.fonts[f_num]
        if  f_tmp then
            local f_tmp_fontname = f_tmp.fontname
            if  f_tmp_fontname then
                local f_id_tmp       = font.getfont(f_num)
                local f_fontname_tmp = f_id_tmp.fontname
                local f_filename_tmp = f_id_tmp.filename
                if  not tbl_fonts_used[f_fontname_tmp] then
                    tbl_fonts_used[f_fontname_tmp] = {f_filename_tmp, f_id_tmp}
                end
            end
        end
    end

    for f_fontname, v in pairs(tbl_fonts_used) do
        f_filename = v[1]
        f_id = v[2]
        if  not tbl_fonts_chars[f_fontname] then
```

```
            tbl_fonts_chars[f_fontname] = { }
            tbl_fonts_chars_init[f_fontname] = { }
            tbl_fonts_chars_medi[f_fontname] = { }
            tbl_fonts_chars_fina[f_fontname] = { }
            local f = fontloader.open(f_filename)
            local char_name
            local char_unicode
            local char_class
            for k, v in pairs(range_tble) do
                for glyph_idx = k, v do
                    if  f_id.characters[glyph_idx] then
                        char_name    = f.glyphs[f_id.characters[glyph_idx].index].name
                        char_unicode = f.glyphs[f_id.characters[glyph_idx].index].unicode
                        char_class   = f.glyphs[f_id.characters[glyph_idx].index].class

                        kashida_fontfamily = token.get_macro("l_texnegar_kashida_fontfamily_
                        fontfamily_match = string.match(f_fontname, "^(" .. kashida_fontfami
                        if fontfamily_match == kashida_fontfamily then
                            if  not tbl_fonts_chars[f_fontname][glyph_idx] then
                                if  string.match(f_fontname, "^(Amiri).*") == "Amiri" and ch
                                    current_kashida_unicode = glyph_idx
                                end
                                tbl_fonts_chars[f_fontname][glyph_idx] = {char_name, char_un
                                for _, pattern in ipairs( pattern_list ) do
                                    local pos_alt = string.match(char_name, pattern)
                                    if  pos_alt == 'ini' or pos_alt == 'AltIni' then
                                        tbl_fonts_chars_init[f_fontname][glyph_idx] = {char_
                                    elseif pos_alt == 'med' or pos_alt == 'AltMed' then
                                        tbl_fonts_chars_medi[f_fontname][glyph_idx] = {char_
                                    elseif pos_alt == 'fin' or pos_alt == 'AltFin' then
                                        tbl_fonts_chars_fina[f_fontname][glyph_idx] = {char_
                                    end
                                end
                            end
                        end
                    end
                end
            end
            fontloader.close(f)
        end
    end
    return tbl_fonts_used, tbl_fonts_chars, tbl_fonts_chars_init, tbl_fonts_chars_medi, tbl_
end

dofile(kpse.find_file("texnegar-ini.lua"))
--
--
-- End of file `texnegar.lua'.
⟨/texnegar-lua⟩
```

1.9 File: texnegar-ini.lua

```
⟨*texnegar-ini-lua⟩
--
-- This is file `texnegar-ini.lua',
-- generated with the docstrip utility.
--
-- The original source files were:
--
-- texnegar.dtx  (with options: ‘texnegar-ini-lua’)
--
-- Copyright (C) 2020-2021 Hossein Movahhedian
--
-- It may be distributed and/or modified under the LaTeX Project Public License,
-- version 1.3c or higher (your choice). The latest version of
-- this license is at: http://www.latex-project.org/lppl.txt
--
-- texnegar_ini        = texnegar_ini or {}
-- local texnegar_ini  = texnegar_ini
-- texnegar_ini.module = {
--     name            = "texnegar_ini",
--     version         = "0.1e",
--     date            = "2021-02-09",
--     description     = "Full implementation of kashida feature in XeLaTex and LuaLaTeX",
--     author          = "Hossein Movahhedian",
--     copyright       = "Hossein Movahhedian",
--     license         = "LPPL v1.3c"
-- }
--
-- -- ^^A%%  texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
-- local err, warn, info, log = luatexbase.provides_module(texnegar_ini.module)
-- texnegar_ini.log     = log  or (function (s) luatexbase.module_info("texnegar_ini", s)
-- texnegar_ini.warning = warn or (function (s) luatexbase.module_warning("texnegar_ini", s)
-- texnegar_ini.error   = err  or (function (s) luatexbase.module_error("texnegar_ini", s)

c_true_bool  = token.create("c_true_bool")

l_texnegar_color_bool                 = token.create("l_texnegar_color_bool")

if  l_texnegar_color_bool.mode == c_true_bool.mode then
    color_tbl = color_tbl or {}
    for item in l_texnegar_color_rgb_tl:gmatch("([^,%s]+)") do
        table.insert(color_tbl, item)
    end
end

dofile(kpse.find_file("texnegar-luatex-kashida.lua"))
--
--
-- End of file ‘texnegar-ini.lua’.
```

⟨/texnegar-ini-lua⟩

1.10 File: texnegar-luatex-kashida.lua

⟨*texnegar-luatex-kashida-lua⟩

```
--
-- This is file ‘texnegar-luatex-kashida.lua’,
-- generated with the docstrip utility.
--
-- The original source files were:
```

```
--
-- texnegar.dtx  (with options: ‘texnegar-luatex-kashida-lua’)
--
-- Copyright (C) 2020-2021 Hossein Movahhedian
--
-- It may be distributed and/or modified under the LaTeX Project Public License,
-- version 1.3c or higher (your choice). The latest version of
-- this license is at: http://www.latex-project.org/lppl.txt
--
-- texnegar_luatex_kashida        = texnegar_luatex_kashida or {}
-- local texnegar_luatex_kashida  = texnegar_luatex_kashida
-- texnegar_luatex_kashida.module = {
--     name                       = "texnegar_luatex_kashida",
--     version                    = "0.1e",
--     date                       = "2021-02-09",
--     description                = "Full implementation of kashida feature in XeLaTex and L
--     author                     = "Hossein Movahhedian",
--     copyright                  = "Hossein Movahhedian",
--     license                    = "LPPL v1.3c"
-- }
--
-- -- ^^A%%  texnegar-lua.dtx -- part of TEXNEGAR <bitbucket.org/dma8hm1334/texnegar>
-- local err, warn, info, log = luatexbase.provides_module(texnegar_luatex_kashida.module)
-- texnegar_luatex_kashida.log     = log  or (function (s) luatexbase.module_info("texnegar_
-- texnegar_luatex_kashida.warning = warn or (function (s) luatexbase.module_warning("texneg
-- texnegar_luatex_kashida.error   = err  or (function (s) luatexbase.module_error("texnegar

local peCharTableInitial, peCharTableMedial, peCharTableFinal, peCharTableDiacritic = dofile
char-table.lua"))

local kashida_unicode = 1600
local kashida_subtype = 256

local COLORSTACK = node.subtype("pdf_colorstack")
local node_id    = node.id
local GLUE       = node_id("glue")
local GLYPH      = node_id("glyph")
local HLIST      = node_id("hlist")
local RULE       = node_id("rule")
local VLIST      = node_id("vlist")
local WHATSIT    = node_id("whatsit")

local l_texnegar_kashida_glyph_bool         = token.create("l_texnegar_kashida_glyph_bool")
local l_texnegar_kashida_leaders_glyph_bool = token.create("l_texnegar_kashida_leaders_glyph
local l_texnegar_kashida_leaders_hrule_bool = token.create("l_texnegar_kashida_leaders_hrule

local l_texnegar_hboxrecursion_bool         = token.create("l_texnegar_hboxrecursion_bool")
local l_texnegar_vboxrecursion_bool         = token.create("l_texnegar_vboxrecursion_bool")

local selected_font = font.current()
local selected_font_old = selected_font

local string_format = string.format
local debug_getinfo = debug.getinfo
```

```

function GetGlyphDimensions(font_file, glyph_index)
    local funcName    = debug_getinfo(1).name
    local funcNparams = debug_getinfo(1).nparams

    local fnt = fontloader.open(font_file)
    local idx = 0
    local fnt_glyphcnt = fnt.glyphcnt
    local fnt_glyphmin = fnt.glyphmin
    local fnt_glyphmax = fnt.glyphmax
    if  fnt_glyphcnt > 0 then
        for idx = fnt_glyphmin, fnt_glyphmax do
            local gl = fnt.glyphs[idx]
            if  gl then
                local gl_unicode = gl.unicode
                if  gl_unicode == glyph_index then
                    local gl_name    = gl.name
                    gl_width   = gl.width
                    local gl_bbox    = gl.boundingbox
                    gl_llx     = gl_bbox[1]
                    gl_depth   = gl_bbox[2]
                    gl_urx     = gl_bbox[3]
                    gl_height  = gl_bbox[4]
                    break
                end
            end
            idx = idx + 1
        end
    end
    fontloader.close(fnt)
    return {width = gl_width, height = gl_height, depth = gl_depth, llx = gl_llx, urx = gl_u
end

function GetGlue(t_plb_line_glue_node, t_plb_node)
    local funcName    = debug_getinfo(1).name
    local funcNparams = debug_getinfo(1).nparams

    local glue_id            = t_plb_line_glue_node.id
    local glue_subtype       = t_plb_line_glue_node.subtype
    local glue_width         = t_plb_line_glue_node.width
    local glue_stretch       = t_plb_line_glue_node.stretch
    local glue_shrink        = t_plb_line_glue_node.shrink
    local eff_glue_width     = node.effective_glue(t_plb_line_glue_node, t_plb_node)
    local glue_stretch_order = t_plb_line_glue_node.stretch_order
    local glue_shrink_order  = t_plb_line_glue_node.shrink_order
    local glue_delta         = 0
    glue_delta = eff_glue_width - glue_width
    return { id = glue_id, subtype = glue_subtype, width = glue_width, stretch = glue_stretc
             shrink = glue_shrink, stretch_order = glue_stretch_order, shrink_order = glue_s
             effective_glue = eff_glue_width, delta = glue_delta }
end

function GetGlyph(t_plb_line_glyph_node, t_tbl_line_fields, t_CharTableInitial, t_CharTableM
    local funcName    = debug_getinfo(1).name
    local funcNparams = debug_getinfo(1).nparams

    local glyph_id      = t_plb_line_glyph_node.id
    local glyph_subtype = t_plb_line_glyph_node.subtype
    local glyph_char    = t_plb_line_glyph_node.char
    local glyph_font    = t_plb_line_glyph_node.font
    local glyph_lang    = t_plb_line_glyph_node.lang
    local glyph_width   = t_plb_line_glyph_node.width
    local glyph_data    = t_plb_line_glyph_node.data

    if  not (t_CharTableInitial[glyph_char] == nil) then
        t_tbl_line_fields.joinerCharInitial = t_tbl_line_fields.joinerCharInitial + 1
        t_plb_line_glyph_node.data = 1
    elseif not (t_CharTableMedial[glyph_char] == nil) then
        t_tbl_line_fields.joinerCharMedial = t_tbl_line_fields.joinerCharMedial + 1
        t_plb_line_glyph_node.data = 2
    elseif not (t_CharTableFinal[glyph_char] == nil) then
        t_tbl_line_fields.joinerCharFinal = t_tbl_line_fields.joinerCharFinal + 1
        t_plb_line_glyph_node.data = 3
    end
    return { id = glyph_id, subtype = glyph_subtype, char = glyph_char, font = glyph_font, l
end

function ProcessTableKashidaHlist(ksh_hlistNode, hbox_num, in_font)
    local funcName    = debug_getinfo(1).name
    local funcNparams = debug_getinfo(1).nparams

    local ksh_hlistNode_id      = ksh_hlistNode.id
    local ksh_hlistNode_subtype = ksh_hlistNode.subtype

    for tn in node.traverse(ksh_hlistNode.head) do
        local tn_id = tn.id
        local tn_subtype = tn.subtype

        if  tn_id == HLIST then
            for tp in node.traverse(tn.head) do
                local tp_id = tp.id
                local tp_subtype = tp.subtype
                if  tp_id == GLYPH then
                    if  l_texnegar_color_bool.mode == c_true_bool.mode then
                        local col_str      = color_tbl[1] .. " " .. color_tbl[2] .. " " .. c
                        local col_str_rg   = col_str .. " rg "
                        local col_str_RG   = col_str .. " RG"

                        local color_push   = node.new(WHATSIT, COLORSTACK)
                        local color_pop    = node.new(WHATSIT, COLORSTACK)
                        color_push.stack   = 0
                        color_pop.stack    = 0
                        color_push.command = 1
                        color_pop.command  = 2
                        glue_ratio         = .2
                        color_push.data       = col_str_rg .. col_str_RG
                        color_pop.data        = col_str_rg .. col_str_RG
                        tn.head = node.insert_before(tn.list, tn.head, node.copy(color_push)
                        tn.head = node.insert_after(tn.list, node.tail(tn.head), node.copy(c
                    end

                    local tp_font = tp.font
                    local tp_char = tp.char
                    tp.font = in_font

                    local ksh_unicode
                    ksh_unicode = font.getfont(in_font).resources.unicodes[’kashida’]
                    if  hbox_num == ’l_texnegar_k_box’ then
                        tp.char = current_kashida_unicode or kashida_unicode
                    elseif hbox_num == ’l_texnegar_ksh_box’ then
                        tp.char = ksh_unicode
                        tn_width = tn.width
                        ksh_hlistNode.width = tn_width
                    end
                elseif  tp_id == HLIST then
                    if  tp.subtype ~= 3 then
                        tbl_kashida_hlist_nodes[ #tbl_kashida_hlist_nodes + 1 ] = tp
                    end
                end
            end
        elseif tn_id == VLIST then
            do end
        elseif tn_id == WHATSIT then
            do end
        elseif  tn_id == GLYPH then
            if  l_texnegar_color_bool.mode == c_true_bool.mode then
                local col_str      = color_tbl[1] .. " " .. color_tbl[2] .. " " .. color_tbl
                local col_str_rg   = col_str .. " rg "
                local col_str_RG   = col_str .. " RG"

                local color_push   = node.new(WHATSIT, COLORSTACK)
                local color_pop    = node.new(WHATSIT, COLORSTACK)
                color_push.stack   = 0
                color_pop.stack    = 0
                color_push.command = 1
                color_pop.command  = 2
                glue_ratio         = .2
                color_push.data       = col_str_rg .. col_str_RG
                color_pop.data        = col_str_rg .. col_str_RG
                ksh_hlistNode.head = node.insert_before(ksh_hlistNode.list, ksh_hlistNode.he
                ksh_hlistNode.head = node.insert_after(ksh_hlistNode.list, node.tail(ksh_hli
            end

            local tn_font = tn.font
            local tn_char = tn.char
            tn.font = in_font

            local ksh_unicode
            ksh_unicode = font.getfont(in_font).resources.unicodes[’kashida’]
            if  hbox_num == ’l_texnegar_k_box’ then
                tn.char = kashida_unicode
            elseif hbox_num == ’l_texnegar_ksh_box’ then
```

```
                tn.char = ksh_unicode
                tn_width = tn.width
                ksh_hlistNode.width = tn_width
            end
        else
           print(string_format("\n tn. Not processed node id is: %d", tn_id))
        end
    end
end

function SetFontInHbox(hbox_num, font_num)
    local funcName    = debug_getinfo(1).name
    local funcNparams = debug_getinfo(1).nparams

    tbl_kashida_hlist_nodes = {}

    local tmp_node
    tmp_node = node.new("hlist")
    tmp_node = tex.getbox(hbox_num)

    ProcessTableKashidaHlist(tmp_node, hbox_num, font_num)

    ::kashida_hlist_BEGIN::
    if  #tbl_kashida_hlist_nodes > 0 then
        local kashida_hlistNodeAdded = table.remove(tbl_kashida_hlist_nodes,1)
        ProcessTableKashidaHlist(kashida_hlistNodeAdded, hbox_num, font_num)
        goto kashida_hlist_BEGIN
    end
end

function StretchGlyph(t_plb_node, t_plb_glyph_node, t_gluePerJoiner, t_dir, t_filler)
    local funcName    = debug_getinfo(1).name
    local funcNparams = debug_getinfo(1).nparams

    if  t_filler == "resized_kashida" then
        SetFontInHbox('l_texnegar_k_box', selected_font)
    elseif t_filler == "leaders+kashida" then
        SetFontInHbox('l_texnegar_ksh_box', selected_font)
    end

    kashida_node = node.new(GLYPH)
    node_glue    = node.new(GLUE)
    node_rule    = node.new(RULE)
    node_hlist   = node.new(HLIST)

    font_current = selected_font
    font_name    = font.fonts[font_current].fullname
    font_file    = font.fonts[font_current].filename
    kashida_char = font.fonts[font_current].characters[1600]

    kashida_node.subtype = kashida_subtype
    kashida_node.font    = font_current
    if  string.match(font_name, "^(Amiri).*") == "Amiri" then
        kashida_node.char = current_kashida_unicode
    else
        kashida_node.char = kashida_unicode
    end
    kashida_node.lang    = tex.language

    kashida_width  = kashida_node.width
    kashida_height = kashida_node.height
    kashida_depth  = kashida_node.depth

    tbl_gl_dimen = GetGlyphDimensions(font_file, kashida_unicode)
    ksh_width, ksh_height, ksh_depth, ksh_llx, ksh_urx =
        tbl_gl_dimen.width, tbl_gl_dimen.height, tbl_gl_dimen.depth, tbl_gl_dimen.llx, tbl_g

    ratio_width = kashida_width / ksh_width
    leaders_height =  ratio_width * ksh_height
    leaders_depth = - ratio_width * ksh_depth

    node_glue.subtype = 100
    node.setglue(node_glue, t_gluePerJoiner, 0, 0, 0, 0)

    if  t_filler == "resized_kashida" then
        node_glue.leader = node.copy_list(tex.box[’l_texnegar_k_box’])
    elseif t_filler == "leaders+kashida" then
        node_glue.leader = node.copy_list(tex.box[’l_texnegar_ksh_box’])
    elseif t_filler == "leaders+hrule" then
        node_glue.leader = node_rule
    end

    node_glue.leader.subtype = 0
    node_glue.leader.height  = leaders_height
    node_glue.leader.depth   = leaders_depth

    node_glue.leader.dir     = t_dir

    local t_plb_glyph_node_next = t_plb_glyph_node.next
    local t_plb_glyph_node_next_id = t_plb_glyph_node_next.id
    if  not t_plb_glyph_node_next then
        node.insert_after(t_plb_node.list, t_plb_glyph_node, node_glue)
    else
        if  t_plb_glyph_node_next_id == GLYPH then
            local t_plb_glyph_node_next_char = t_plb_glyph_node_next.char
            if  peCharTableDiacritic[t_plb_glyph_node_next_char] then
                node.insert_after(t_plb_node.list, t_plb_glyph_node_next, node_glue)
            else
                node.insert_after(t_plb_node.list, t_plb_glyph_node, node_glue)
            end
        else
            node.insert_after(t_plb_node.list, t_plb_glyph_node, node_glue)
        end
    end
    if  t_filler == "leaders+hrule" then
        for tn in node.traverse(t_plb_node.head) do
            local tn_id = tn.id
            local tn_subtype = tn.subtype

            if  tn_id == GLUE and tn_subtype == 100 then
                local t_hbox = node.new(HLIST)
                local t_hrule = node.copy(tn)

                if  string.match(font_name, "^(Amiri).*") == "Amiri" then
                    t_hrule.leader.height = kashida_height
                    t_hrule.leader.depth  = kashida_depth
                end

                t_hbox.head = node.insert_after(t_hbox.list, t_hbox.head,t_hrule)
                t_plb_node.head = node.insert_after(t_plb_node.list, tn, t_hbox)

                if  l_texnegar_color_bool.mode == c_true_bool.mode then
                    local col_str      = color_tbl[1] .. " " .. color_tbl[2] .. " " .. color
                    local col_str_rg   = col_str .. " rg "
                    local col_str_RG   = col_str .. " RG"

                    local color_push   = node.new(WHATSIT, COLORSTACK)
                    local color_pop    = node.new(WHATSIT, COLORSTACK)
                    color_push.stack   = 0
                    color_pop.stack    = 0
                    color_push.command = 1
                    color_pop.command  = 2
                    glue_ratio         = .2
                    color_push.data       = col_str_rg .. col_str_RG
                    color_pop.data        = col_str_rg .. col_str_RG
                    t_hbox.head = node.insert_before(t_hbox.list, t_hbox.head, node.copy(col
                    t_hbox.head = node.insert_after(t_hbox.list, node.tail(t_hbox.head), nod
                end
            end
        end
    end
end

function GetFillerSpec(t_plb_node, t_plb_head_node, t_tbl_line_fields, t_CharTableInitial, t
    local funcName    = debug_getinfo(1).name
    local funcNparams = debug_getinfo(1).nparams

    t_plb_node_id = t_plb_node.id
    t_plb_node_subtype = t_plb_node.subtype

    for p in node.traverse(t_plb_head_node) do
        local p_id = p.id
        local p_subtype = p.subtype
        if  p_id == HLIST then
            t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - p.
            if  p.subtype ~= 3 then
                tbl_hlist_nodes[ #tbl_hlist_nodes + 1 ] = p
            end
        elseif p_id == VLIST then
            t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - p.
            tbl_vlist_nodes[ #tbl_vlist_nodes + 1 ] = p
        elseif p_id == GLUE then
```

```
            tbl_p_glue = GetGlue(p, t_plb_node)
            t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - tb
            t_tbl_line_fields.total_glues = t_tbl_line_fields.total_glues + 1
            t_tbl_line_fields.stretchedGlue = t_tbl_line_fields.stretchedGlue + tbl_p_glue["
        elseif p_id == GLYPH then
            tbl_p_glyph, t_tbl_line_fields = GetGlyph(p, t_tbl_line_fields, t_CharTableIniti
            selected_font_old = selected_font
            selected_font = tbl_p_glyph["font"]
            t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - tb
            t_tbl_line_fields.total_glyphs = t_tbl_line_fields.total_glyphs + 1
        end
    end

    t_tbl_line_fields.total_joiners = t_tbl_line_fields.joinerCharInitial + t_tbl_line_field
    t_tbl_line_fields.gluePerJoiner = 0
    if  t_tbl_line_fields.total_glues == 0 then
        t_tbl_line_fields.stretchedGlue = t_tbl_line_fields.lineWidthRemainder
    end
    if  t_tbl_line_fields.total_joiners > 0 then
        t_tbl_line_fields.gluePerJoiner           = t_tbl_line_fields.stretchedGlue // t_tbl
        t_tbl_line_fields.stretchedGlueRemaineder = t_tbl_line_fields.stretchedGlue % t_tbl_
    elseif t_tbl_line_fields.total_joiners == 1 then
        t_tbl_line_fields.gluePerJoiner = t_tbl_line_fields.stretchedGlue
    end

    return t_tbl_line_fields
end

function ProcessTableHlist(tmphl_n)
    local funcName    = debug_getinfo(1).name
    local funcNparams = debug_getinfo(1).nparams

    local tmphl_n_id      = tmphl_n.id
    local tmphl_n_subtype = tmphl_n.subtype

    local tbl_line_fields = { line_dir          = "", line_width       = 0, lineWidthRemaind
                              joinerCharInitial = 0,  joinerCharMedial = 0, joinerCharFinal
                              stretchedGlue     = 0,  total_glues      = 0, gluePerJoiner

    local tbl_p_glue, tbl_p_glyph

    if  (tmphl_n_id == HLIST) and (tmphl_n_subtype == 1 or tmphl_n_subtype == 2) then
        tbl_line_fields.line_width = tmphl_n.width
        tbl_line_fields.line_dir   = tmphl_n.dir
        tbl_line_fields.lineWidthRemainder = tbl_line_fields.line_width

        if  tbl_line_fields.line_dir == "TLT" then
            tbl_line_fields = GetFillerSpec(tmphl_n, tmphl_n.head, tbl_line_fields, peCharTa

            if  tbl_line_fields.total_joiners == 0 or tbl_line_fields.gluePerJoiner == 0 or
                goto continue
            end

            for q in node.traverse_id(GLUE, tmphl_n.head) do
```

```
                local eff_glue_width     = node.effective_glue(q, tmphl_n)
                node.setglue(q, q.width, 0, 0, q.stretch_order, q.glue_shrink_order)
            end

            for r in node.traverse_id(GLYPH, tmphl_n.head) do
                local r_data = r.data
                if  r_data == 1 or r.data == 2 then
                    StretchGlyph(tmphl_n, r, tbl_line_fields.gluePerJoiner, tbl_line_fields.
                elseif r.data == 3 then
                    goto for_loop_01
                end
                ::for_loop_01::
            end
            tbl_line_fields.line_width = tmphl_n.width
            tbl_line_fields.lineWidthRemainder = line_width
        elseif tbl_line_fields.line_dir == "TRT" then
            tbl_line_fields = GetFillerSpec(tmphl_n, tmphl_n.head, tbl_line_fields, peCharTa
            if  tbl_line_fields.total_joiners == 0 or tbl_line_fields.gluePerJoiner == 0 or
                goto continue
            end

            for q in node.traverse_id(GLUE, tmphl_n.head) do
                local eff_glue_width     = node.effective_glue(q, tmphl_n)
                node.setglue(q, q.width, 0, 0, q.stretch_order, q.glue_shrink_order)
            end

            for r in node.traverse_id(GLYPH, tmphl_n.head) do
                local r_data = r.data
                if  r_data == 1 or r.data == 2 then
                    StretchGlyph(tmphl_n, r, tbl_line_fields.gluePerJoiner, tbl_line_fields.
                elseif r.data == 3 then
                    goto for_loop_02
                end
                ::for_loop_02::
            end
            tbl_line_fields.line_width = tmphl_n.width
            tbl_line_fields.lineWidthRemainder = line_width
        else
            print(string_format("\n Line direction ’%s’ is not supported yet!", tbl_line_fie
        end
    end
    ::continue::
end

function ProcessTableVlist(tmpvl_n)
    local funcName    = debug_getinfo(1).name
    local funcNparams = debug_getinfo(1).nparams

    local tmpvl_n_id      = tmpvl_n.id
    local tmpvl_n_subtype = tmpvl_n.subtype

    for vbNode in node.traverse(tmpvl_n) do
        if  vbNode.id == VLIST and vbNode.subtype == 0 then
            for tr_vbNode in node.traverse(vbNode.head) do
```

```
                if  (tr_vbNode.id == HLIST) and (tr_vbNode.subtype == 1 or tr_vbNode.subtype
                    ProcessTableHlist(tr_vbNode)
                end
            end
        end
    end
end

function PostLineBreakFilter(hboxes_stack, groupcode)
    local funcName    = debug_getinfo(1).name
    local funcNparams = debug_getinfo(1).nparams

    funcName = "PostLineBreakFilter"

    local tbl_fonts_used = { }
    local tbl_fonts_chars = { }
    local tbl_fonts_chars_init = { }
    local tbl_fonts_chars_medi = { }
    local tbl_fonts_chars_fina = { }

    tbl_fonts_used, tbl_fonts_chars, tbl_fonts_chars_init, tbl_fonts_chars_medi, tbl_fonts_c

    local f_fontname

    for f_fontname, v in pairs(tbl_fonts_used) do
        for k1, v1 in pairs(tbl_fonts_chars_init[f_fontname]) do
            if  k1 and not peCharTableInitial[k1] then
                peCharTableInitial[k1] = utf8.char(k1)
            end
        end

        for k1, v1 in pairs(tbl_fonts_chars_medi[f_fontname]) do
            if  k1 and not peCharTableMedial[k1] then
                peCharTableMedial[k1] = utf8.char(k1)
            end
        end

        for k1, v1 in pairs(tbl_fonts_chars_fina[f_fontname]) do
            if  k1 and not peCharTableFinal[k1] then
                peCharTableFinal[k1] = utf8.char(k1)
            end
        end
    end

    tbl_hlist_nodes = {}
    tbl_vlist_nodes = {}
    for hlistNode in node.traverse(hboxes_stack) do
        if  node.next(hlistNode) == nil then
            goto END
        end

        ProcessTableHlist(hlistNode)

        if   l_texnegar_hboxrecursion_bool.mode == c_true_bool.mode then
```

```
            ::hboxBEGIN::
            if  #tbl_hlist_nodes > 0 then
                local hlistNodeAdded = table.remove(tbl_hlist_nodes,1)
                ProcessTableHlist(hlistNodeAdded)
                goto hboxBEGIN
            end
        end

        if   l_texnegar_vboxrecursion_bool.mode == c_true_bool.mode then
            ::vboxBEGIN::
            if  #tbl_vlist_nodes > 0 then
                local vlistNodeAdded = table.remove(tbl_vlist_nodes,1)
                ProcessTableVlist(vlistNodeAdded)
                goto vboxBEGIN
            end
        end

        ::END::
    end
    return hboxes_stack
end

if  l_texnegar_kashida_glyph_bool.mode == c_true_bool.mode then
    filler_pe = "resized_kashida"
elseif l_texnegar_kashida_leaders_glyph_bool.mode == c_true_bool.mode then
     filler_pe = "leaders+kashida"
elseif l_texnegar_kashida_leaders_hrule_bool.mode == c_true_bool.mode then
    filler_pe = "leaders+hrule"
else
    print(string_format" Unknown kashida value.")
end

function StartStretching()
    if  not luatexbase.in_callback(’post_linebreak_filter’, ’insertKashida’) then
        luatexbase.add_to_callback(’post_linebreak_filter’, PostLineBreakFilter, ’insertKash
    end
end

function StopStretching()
    if  luatexbase.in_callback(’post_linebreak_filter’, ’insertKashida’) then
        luatexbase.remove_from_callback(’post_linebreak_filter’, ’insertKashida’)
    end
end
--
--
-- End of file ‘texnegar-luatex-kashida.lua’.
⟨/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 refered 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 implementaion 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 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 Ierusalimschy, *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 2ε 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, XeTeX reference guide, `CTAN:info/xetexref/xetex-reference.pdf`, 2019-12-09

[13] Package `xetex`: Jonathan Kew, About XeTeX, `CTAN:systems/doc/xetex/XeTeX-notes.pdf`, 2005-10-17

[14] Package `xetex`: Michel Goossens, The XeTeX Companion, `http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf`, 2009-08-19

[15] Website: Stack Exchange: Hot Questions, TeX-LaTeX Q&A for users of TeX, LaTeX, ConTeXt, and related typesetting systems, `tex.stackexchange.com`

[16] Website: LuaTeX Wiki, LuaTeX 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.