The TEXNegar package
Kashida justification in LuaTeX and XeTeX
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

Contents

1 TEXNegar Implementation

1.1 File: texnegar.sty

```
⟨*texnegar-sty⟩
\RequirePackage{xparse}
\RequirePackage{l3keys2e}
\RequirePackage{graphicx}[2019-11-30]
\RequirePackage[dvipsnames,svgnames,x11names]{xcolor}[2016/05/11]
\ProvidesExplPackage {texnegar} {2020-08-30} {0.1b} { 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} {2020-08-30} {0.1b} { 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} {2020-08-30} {0.1b} { 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} {2020-08-30} {0.1b} { 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_kashida_fix_bool

\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 }
  {
    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
          \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

 \endinput
⟨/texnegar-ini-tex⟩
```

1.5 File: texnegar-common-kashida.tex

```
⟨*texnegar-common-kashida-tex⟩
\ProvidesExplFile {texnegar-common-kashida.tex} {2020-08-30} {0.1b} { 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 }
   }
```

```
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_li
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
```

```
        }
      }
    \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 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                = "0.1b",
--     date                   = "2020-08-30",
--     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 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
--
--
-- 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 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       = "0.1b",
--     date          = "2020-08-30",
--     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)

tex.enableprimitives('',tex.extraprimitives ())
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 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         = "0.1b",
--     date            = "2020-08-30",
--     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"))   -- leaders + resized glyph U+0640
--
--
-- 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 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                    = "0.1b",
--     date                       = "2020-08-30",
--     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 = dofile(kpse.find_file("texne
char-table.lua"))

local kashida_unicode = 1600
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_hrule_bool = token.create("l_texnegar_kashida_leaders_hrule
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                     gl_width   = gl.width
1038                     local gl_bbox    = gl.boundingbox
1039                     gl_llx     = gl_bbox[1]
1040                     gl_depth   = gl_bbox[2]
1041                     gl_urx     = gl_bbox[3]
1042                     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_u
1051 end
1052
```

```
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 == 0 then
            for tp in node.traverse(tn.head) do
                local tp_id = tp.id
                local tp_subtype = tp.subtype
                if  tp_id == 29 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 = 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 == 0 then
                    if  tp.subtype ~= 3 then
                        tbl_kashida_hlist_nodes[ #tbl_kashida_hlist_nodes + 1 ] = tp
                    end
                end
            end
        elseif tn_id == 1 then
            do end
        elseif tn_id == 8 then
            do end
        elseif  tn_id == 29 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
    kashida_node.char    = kashida_unicode
    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

    node.insert_after(t_plb_node.list, t_plb_glyph_node, node_glue)
    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 == 12 and tn_subtype == 100 then
                local t_hbox = node.new(HLIST)
                local t_hrule = node.copy(tn)
                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 == 0 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 == 1 then
            t_tbl_line_fields.lineWidthRemainder = t_tbl_line_fields.lineWidthRemainder - p.
            tbl_vlist_nodes[ #tbl_vlist_nodes + 1 ] = p
        elseif p_id == 12 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 == 29 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 == 0) 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

    print(string_format(" %s: 00-0 tmpvl_n: id: %d, subtype: %d", funcName, tmpvl_n_id, tmpv
    for vbNode in node.traverse(tmpvl_n) do
        if  vbNode.id == 1 and vbNode.subtype == 0 then
            for tr_vbNode in node.traverse(vbNode.head) do
                if  (tr_vbNode.id == 0) 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

    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.

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.

U

X

Z