作りましょう 0.7

パラメタ方式フォントファミリ
校とプリティプリントのソース

Tsukurimashou 0.7

Parametric Font Family
Proofs and pretty-printed source code

Matthew Skala
mskala@ansuz.sooke.bc.ca

2013年3月7日　　March 7, 2013

Proofs and pretty-printed source code for Tsukurimashou

Contents

I Infrastructure 13

IV U+0000 to U+2FFF 845

V U+3000 to U+4DFF 1481

VI U+4E00 to U+61FF 1757

VII U+6200 to U+75FF 2227

IX U+8A00 to U+9FFF 2683

X U+A000 to U+10FFFF 2887

XI Jieubsida core 3197

Volume I

Infrastructure

preintro.mp	PRE
tsuku-bk.mp	BK
tsuku-kg.mp	KG
tsuku-mg.mp	MG
tsuku-mi.mp	MI
tsuku-ps.mp	PS
tsuku-bq.mp	BQ
tsuku-dq.mp	DQ
tsuku-el.mp	EL
tsuku-eq.mp	EQ
tsuku-lw.mp	LW
intro.mp	INTR
fntbase.mp	FNTB
obstack.mp	OBST
frac-intro.mp	FRAC
latin-intro.mp	LATI
accent.mp	ACCE
bcircle.mp	BCIR
buildkanji.mp	BUIL
dakuten.mp	DAKU
enclosed.mp	ENCL
genjimon.mp	GENJ
hiragana.mp	HIRA
iching.mp	ICHI
katakana.mp	KATA
latin.mp	LATI
numerals.mp	NUME
ogonek.mp	OGON
punct.mp	PUNC
serif.mp	SERI

preintro.mp

```
%
% Early shared code for Tsukurimashou
% Copyright (C) 2011, 2012 Matthew Skala
%
% This program is free software: you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation, version 3.
%
% As a special exception, if you create a document which uses this font, and
% embed this font or unaltered portions of this font into the document, this
% font does not by itself cause the resulting document to be covered by the
% GNU General Public License. This exception does not however invalidate any
% other reasons why the document might be covered by the GNU General Public
% License. If you modify this font, you may extend this exception to your
% version of the font, but you are not obligated to do so. If you do not
% wish to do so, delete this exception statement from your version.
%
% This program is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
% GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along with this program. If not, see <http://www.gnu.org/licenses/>.
%
% Matthew Skala
% http://ansuz.sooke.bc.ca/
% mskala@ansuz.sooke.bc.ca
%
```

Infrastructure

```
% INFRASTRUCTURE

% When I say nonstopmode I mean nonstopmode, dammit!
nonstopmode;
def errorstopmode = nonstopmode enddef;

% no chars we don't define, please
no_implicit_spaces:=1;

% load library from METATYPE1
input fntbase.mp;

```

```
% file inclusion gatekeeper
vardef inclusion_lock(suffix fn) =
  if known already_included.fn:
    endinput;
  fi;
  boolean already_included.fn;
  already_included.fn:=true;
enddef;

% late inclusion
numeric late_include_count;
late_include_count:=0;
string late_include[];

vardef include_late(expr fn) =
  late_include_count:=late_include_count+1;
  late_include[late_include_count]:=fn;
enddef;

vardef do_late_includes =
  for i:=1 upto late_include_count:
    scantokens (”input ” & late_include[i]);
  endfor;
  late_include_count:=0;
enddef;

```

Font Parameter Defaults

```
% FONT PARAMETER DEFAULTS

% basic brush definition
transform tsu_brush_xf;
tsu_brush_shape:=1.0;
tsu_brush_angle:=0;

% special brush for punctuation
tsu_pbrush_size:=50;
tsu_pbrush_shape:=1.0;
tsu_pbrush_angle:=0;
tsu_punct_size:=100;

% size the handakuten
handakuten_inner:=120;
handakuten_outer:=200;

```

```
% general shape tweaker
mincho:=0;

% slant during rescaling, for italics
rescale_slant:=0;

% control appearance of corners
boolean sharp_corners;
sharp_corners:=false;

% for naming the font
string familyname,stylename;
familyname:=”Tsukurimashou”;
stylename:=””;

% brush option override
def tsu_brush_opt(expr n,l) = nib(n)(l) enddef;

% bo_serif type; point lp; direction lp; brush tip size
vardef tsu_serif.choose(expr bst,plp,dlp,l,bts,bos) =
enddef;

% do ”modern” width alternation
boolean do_alternation;
do_alternation:=false;

% handle outline mode for Genjimon
boolean genji_outline;
genji_outline:=false;

% prepare to detect proportional spacing
boolean is_proportional;
is_proportional:=false;

% prepare to detect blackletter
boolean is_blackletter;
is_blackletter:=false;

% prepare to detect fine IDCs
boolean fine_idcs;
fine_idcs:=false;

% prepare to detect italic hook shapes
boolean do_italic_hook;
do_italic_hook:=false;
```

tsuku-bk.mp

```
%
% Tsukurimashou Bokukko
% Copyright (C) 2011 Matthew Skala
%
[Standard copyright notice]

% TSUKURIMASHOU BOKUKKO

input preintro.mp;

stylename:=”Bokukko”;

mincho:=0.3;

(0,4) transformed tsu_brush_xf = (0.8,0.95);
(1,1) transformed tsu_brush_xf = (1.02,0.80);
(4,0) transformed tsu_brush_xf = (3.8,0.95);

tsu_brush_min:=0.80;
tsu_brush_max:=0.95;
tsu_brush_shape:=0.3;
tsu_brush_angle:=20;

tsu_pbrush_size:=60;
tsu_pbrush_shape:=0.3;
tsu_pbrush_angle:=20;

def tsu_brush_opt(expr n,l) = cut(n,rel 120)(l) enddef;
sharp_corners:=true;

genji_outline:=true;
genji_hw:=0.55;

input intro.mp;
```

tsuku-kg.mp

```
%
% Tsukurimashou Kaku
% Copyright (C) 2011 Matthew Skala
%
[Standard copyright notice]

% TSUKURIMASHOU KAKU

input preintro.mp;

stylename:=”Kaku”;

(0,4) transformed tsu_brush_xf = (4,0.75);
(1,1) transformed tsu_brush_xf = (1,0.62);
(4,0) transformed tsu_brush_xf = (0,0.75);

tsu_brush_min:=0.62;
tsu_brush_max:=0.75;

def tsu_brush_opt(expr n,l) = cut(n,rel 90)(l) enddef;
sharp_corners:=true;

input intro.mp;
```

tsuku-mg.mp

```
%
% Tsukurimashou Maru
% Copyright (C) 2011 Matthew Skala
%
[Standard copyright notice]

% TSUKURIMASHOU MARU

input preintro.mp;

stylename:=”Maru”;

(0,4) transformed tsu_brush_xf = (4,0.74);
(1,1) transformed tsu_brush_xf = (1,0.65);
(4,0) transformed tsu_brush_xf = (0,0.74);

tsu_brush_min:=0.65;
tsu_brush_max:=0.74;

input intro.mp;
```

tsuku-mi.mp

```
%
% Tsukurimashou Mincho
% Copyright (C) 2011 Matthew Skala
%
[Standard copyright notice]

% TSUKURIMASHOU MINCHO

input preintro.mp;

stylename:=”Mincho”;

mincho:=1;

(0,4) transformed tsu_brush_xf = (0.0,1.1);
(1,1) transformed tsu_brush_xf = (1.2,0.35);
(4,0) transformed tsu_brush_xf = (4.8,1.1);

tsu_brush_min:=0.35;
tsu_brush_max:=1.05;

tsu_brush_shape:=0.38;
tsu_brush_angle:=1;

tsu_pbrush_size:=60;
tsu_pbrush_shape:=0.38;
tsu_pbrush_angle:=1;

input serif.mp;

for i=1 upto 10:
  tsu_do_serif[i]:=true;
endfor;

do_alternation:=true;

genji_outline:=true;
genji_hw:=0.2;

input intro.mp;
```

tsuku-ps.mp

```
%
% Proportional spacing modifications for Tsukurimashou
% Copyright (C) 2011 Matthew Skala
%
[Standard copyright notice]



vardef tsu_rescale_half = tsu_rescale_full; enddef;
vardef tsu_rescale_half_lc = tsu_rescale_full; enddef;
vardef tsu_rescale_half_katakana = tsu_rescale_full; enddef;
vardef tsu_rescale_decenter = tsu_rescale_full; enddef;

is_proportional:=true;

tsu_rescale_full;
```

tsuku-bq.mp

```
%
% Bold font weight
% Copyright (C) 2012 Matthew Skala
%
[Standard copyright notice]

% BOLD WEIGHT

transform weight_xf;
(0,0.62) transformed weight_xf = (0,1.082);
(1,0.62) transformed weight_xf = (1,1.082);
(0,0.75) transformed weight_xf = (0,1.191);

tsu_brush_xf:=tsu_brush_xf transformed weight_xf;
tsu_brush_min:=ypart ((0,tsu_brush_min) transformed weight_xf);
tsu_brush_max:=ypart ((0,tsu_brush_max) transformed weight_xf);

tsu_pbrush_size:=tsu_pbrush_size*(115/100);
tsu_punct_size:=120;

handakuten_outer:=260;

calc_mbrush_size;
```

tsuku-dq.mp

```
%
% Demibold font weight
% Copyright (C) 2012 Matthew Skala
%
[Standard copyright notice]

% DEMIBOLD WEIGHT

transform weight_xf;
(0,0.62) transformed weight_xf = (0,0.781);
(1,0.62) transformed weight_xf = (1,0.781);
(0,0.75) transformed weight_xf = (0,0.945);

tsu_brush_xf:=tsu_brush_xf transformed weight_xf;
tsu_brush_min:=ypart ((0,tsu_brush_min) transformed weight_xf);
tsu_brush_max:=ypart ((0,tsu_brush_max) transformed weight_xf);

tsu_pbrush_size:=tsu_pbrush_size*(115/100);
tsu_punct_size:=110;

handakuten_outer:=260;

calc_mbrush_size;
```

tsuku-el.mp

```
%
% Extra-Light font weight (Tenshi no Kami when added to Maru)
% Copyright (C) 2011, 2012 Matthew Skala
%
[Standard copyright notice]

% EXTRA-LIGHT WEIGHT

transform weight_xf;
(0,0.62) transformed weight_xf = (0,0.15);
(1,0.62) transformed weight_xf = (1,0.15);
(0,0.75) transformed weight_xf = (0,0.15);

tsu_brush_xf:=tsu_brush_xf transformed weight_xf;
tsu_brush_min:=ypart ((0,tsu_brush_min) transformed weight_xf);
tsu_brush_max:=ypart ((0,tsu_brush_max) transformed weight_xf);

tsu_pbrush_size:=tsu_pbrush_size*(15/100);
tsu_punct_size:=80;

handakuten_inner:=170;

fine_idcs:=true;

calc_mbrush_size;
```

tsuku-eq.mp

```
%
% Extra-Bold font weight (Anbiruteki when added to Maru)
% Copyright (C) 2011, 2012 Matthew Skala
%
[Standard copyright notice]

% EXTRA-BOLD WEIGHT

transform weight_xf;
(0,0.62) transformed weight_xf = (0,1.5);
(1,0.62) transformed weight_xf = (1,1.5);
(0,0.75) transformed weight_xf = (0,1.5);

tsu_brush_xf:=tsu_brush_xf transformed weight_xf;
tsu_brush_min:=ypart ((0,tsu_brush_min) transformed weight_xf);
tsu_brush_max:=ypart ((0,tsu_brush_max) transformed weight_xf);

tsu_pbrush_size:=tsu_pbrush_size*(115/100);
tsu_punct_size:=130;

handakuten_outer:=260;

calc_mbrush_size;
```

tsuku-lw.mp

```
%
% Light font weight
% Copyright (C) 2012 Matthew Skala
%
[Standard copyright notice]

% LIGHT WEIGHT

transform weight_xf;
(0,0.62) transformed weight_xf = (0,0.386);
(1,0.62) transformed weight_xf = (1,0.386);
(0,0.75) transformed weight_xf = (0,0.439);

tsu_brush_xf:=tsu_brush_xf transformed weight_xf;
tsu_brush_min:=ypart ((0,tsu_brush_min) transformed weight_xf);
tsu_brush_max:=ypart ((0,tsu_brush_max) transformed weight_xf);

tsu_pbrush_size:=tsu_pbrush_size*(35/100);
tsu_punct_size:=90;

handakuten_inner:=170;

fine_idcs:=true;

calc_mbrush_size;
```

intro.mp

```
%
% General shared code for Tsukurimashou
% Copyright (C) 2011, 2012 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(intro);

```

```

%
% Tsukurimashou intro - utility routines for all pages
%

slang:=0;

pf_info_quad 1000;
pf_info_space 1000, 0, 0;
pf_info_familyname familyname;
pf_info_fontname
  (familyname & stylename & ”Subfont”),
  (familyname & ” ” & stylename & ” Subfont”);
pf_info_fixedpitch true;
pf_info_capheight 900;
pf_info_xheight 585;
pf_info_ascender 985;
pf_info_descender 265;

pair centre_pt;
centre_pt=(500,390);
latin_vcentre:=430;
latin_wide_baseline:=25;
latin_wide_top:=750;
wide_margin:=30;
narrow_margin:=40;

```

```

input bcircle.mp;
input obstack.mp;

```

```

default_nib:=fix_nib(100,100,0);

def begintsuglyph(expr name,code) =
  message name;
  encode(name) (code); standard_introduce(name);
  write ("BEGINGLYPH '"&name& "' "&decimal code) to "proof.prf";
  beginglyph(name);
    init_obstack;
    string perl_structure;
    perl_structure:="$structure{'"&name&"'}=['"&name&"',";
enddef;

def endtsuglyph =
    if rescale_to.right>0:
      fix_hsbw((rescale_to.left+rescale_to.right),0,0);
    else:
      fix_hsbw(0,0,0);
    fi;
  endglyph;
  perl_structure:=perl_structure&"];";
  write "PERL_STRUCTURE "&perl_structure to "proof.prf";
  if rescale_to.right>0:
    write ("ENDGLYPH 0 "&decimal (rescale_to.left+rescale_to.right))
      to "proof.prf";
  else:
    write "ENDGLYPH -10 0" to "proof.prf";
  fi;
  sp:=0;
enddef;

def tsu_brush_tip_size(expr l,q) =
  begingroup
    numeric y,yy;
    y:=ypart (point l of q);
    if y<tsu_brush_min:
      yy:=tsu_brush_min;
    elseif y>tsu_brush_max:
      yy:=tsu_brush_max;
    else:
      yy:=y;
    fi;
    yy
  endgroup
enddef;

def tsu_brush_tip(expr l,p,q,bsi,is_start,is_end,is_alt) =
  begingroup
    numeric y;
    y=tsu_brush_tip_size(l,q);
    if is_alt and do_alternation:
      fix_nib(bsi*y*tsu_brush_shape,bsi*y*tsu_brush_shape,tsu_brush_angle)
    else:
      fix_nib(bsi*y,bsi*y*tsu_brush_shape,tsu_brush_angle)
    fi
  endgroup
enddef;

% rescaling for half/double width
% this is basically global because it will be shared by most glyphs in a file

path width_curve;

def tsu_rescale_full =
  rescale_from.left:=wide_margin;
  rescale_from.right:=1000-wide_margin;
  rescale_from.top:=ypart centre_pt;
  rescale_from.bottom:=latin_wide_baseline;

  rescale_to.left:=wide_margin;
  rescale_to.right:=1000-wide_margin;
  rescale_to.top:=ypart centre_pt;
  rescale_to.bottom:=latin_wide_baseline;

  rescale_skew:=0;

  width_curve:=(-1,-1)--(2000,2000);
enddef;

def tsu_rescale_half =
  rescale_from.left:=wide_margin;
  rescale_from.right:=1000-wide_margin;
  rescale_from.top:=ypart centre_pt;
  rescale_from.bottom:=latin_wide_baseline;

  rescale_to.left:=narrow_margin;
  rescale_to.right:=500-narrow_margin;
  rescale_to.top:=latin_vcentre;
  rescale_to.bottom:=0;

  rescale_skew:=0;

  width_curve:=((-1,-1)--(100,100))..(940,410)..{right}(2000,1000);
enddef;

def tsu_rescale_half_lc =
  rescale_from.left:=wide_margin*3.5;
  rescale_from.right:=1000-wide_margin*3.5;
  rescale_from.top:=ypart centre_pt;
  rescale_from.bottom:=latin_wide_baseline;

  rescale_to.left:=narrow_margin;
  rescale_to.right:=500-narrow_margin;
  rescale_to.top:=latin_vcentre;
  rescale_to.bottom:=0;

  rescale_skew:=0;

  width_curve:=((-1,-1)--(100,100))..(780,410)..{right}(2000,1000);
enddef;

def tsu_rescale_half_katakana =
  rescale_from.left:=wide_margin;
  rescale_from.right:=1000-wide_margin;
  rescale_from.top:=700;
  rescale_from.bottom:=0;

  rescale_to.left:=narrow_margin;
  rescale_to.right:=500-narrow_margin;
  rescale_to.top:=670;
  rescale_to.bottom:=30;

  rescale_skew:=8;

  width_curve:=((-1,-1)--(100,100))..(860,440)..{right}(2000,1000);
enddef;

def tsu_rescale_double =
  rescale_from.left:=narrow_margin;
  rescale_from.right:=500-narrow_margin;
  rescale_from.top:=latin_vcentre;
  rescale_from.bottom:=0;

  rescale_to.left:=wide_margin;
  rescale_to.right:=1000-wide_margin;
  rescale_to.top:=ypart centre_pt;
  rescale_to.bottom:=latin_wide_baseline;

  rescale_skew:=0;

  width_curve:=(-1,-1)--(2000,2000);
enddef;

def tsu_rescale_decenter =
  rescale_from.left:=300;
  rescale_from.right:=700;
  rescale_from.top:=ypart centre_pt;
  rescale_from.bottom:=latin_wide_baseline;

  rescale_to.left:=50;
  rescale_to.right:=450;
  rescale_to.top:=latin_vcentre;
  rescale_to.bottom:=0;

  rescale_skew:=0;

  width_curve:=(-1,-1)--(2000,2000);
enddef;

def tsu_rescale_native_narrow =
  rescale_from.left:=narrow_margin;
  rescale_from.right:=500-narrow_margin;
  rescale_from.top:=latin_vcentre;
  rescale_from.bottom:=0;

  rescale_to.left:=narrow_margin;
  rescale_to.right:=500-narrow_margin;
  rescale_to.top:=latin_vcentre;
  rescale_to.bottom:=0;

  rescale_skew:=0;

  width_curve:=(-1,-1)--(2000,2000);
enddef;

def tsu_rescale_native_zero =
  rescale_from.left:=0;
  rescale_from.right:=0;
  rescale_from.top:=1000;
  rescale_from.bottom:=0;

  rescale_to.left:=0;
  rescale_to.right:=0;
  rescale_to.top:=1000;
  rescale_to.bottom:=0;

  rescale_skew:=0;

  width_curve:=(-1,-1)--(2000,2000);
enddef;

def tsu_rescale_native_conditional =
  if is_proportional:
    tsu_rescale_full;
  else:
    tsu_rescale_native_narrow;
  fi;
enddef;

def tsu_rescale_combining_full =
  rescale_from.left:=wide_margin;
  rescale_from.right:=1000-wide_margin;
  rescale_from.top:=ypart centre_pt;
  rescale_from.bottom:=latin_wide_baseline;

  rescale_to.left:=wide_margin-1000;
  rescale_to.right:=-wide_margin;
  rescale_to.top:=ypart centre_pt;
  rescale_to.bottom:=latin_wide_baseline;

  rescale_skew:=0;

  width_curve:=(-1,-1)--(2000,2000);
enddef;

def tsu_rescale_combining_half =
  rescale_from.left:=wide_margin;
  rescale_from.right:=1000-wide_margin;
  rescale_from.top:=ypart centre_pt;
  rescale_from.bottom:=latin_wide_baseline;

  rescale_to.left:=narrow_margin-500;
  rescale_to.right:=-narrow_margin;
  rescale_to.top:=latin_vcentre;
  rescale_to.bottom:=0;

  rescale_skew:=0;

  width_curve:=((-1,-1)--(100,100))..(940,410)..{right}(2000,1000);
enddef;

def tsu_rescale_combining_accent =
  if is_proportional:
    tsu_rescale_combining_full;
  else:
    tsu_rescale_combining_half;
  fi;
enddef;

tsu_rescale_full;

def tsu_slant_xform =
  begingroup
    save st,cp;
    transform st;
    pair cp;

    cp:=((rescale_from.left+rescale_from.right)/2,rescale_from.bottom);
    cp transformed st=cp;
    cp+(100,0) transformed st=cp+(100,0);
    cp+(0,100) transformed st=cp+(rescale_slant/10,100);
    st
  endgroup
enddef;

def tsu_rescale_xform =
  begingroup
    save t,st,cp;
    transform t,st;
    st:=tsu_slant_xform;
    t:=st;
    % check if rescaling is active
    if (rescale_from.left<>rescale_to.left)
    or (rescale_from.right<>rescale_to.right): begingroup
      save i,xa,xb,lf,rf,wf,lt,rt,wt;
      numeric i,xa,xb,lf,rf,wf,lt,rt,wt;
      transform t;
      % find the bounds of the paths
      if find_stroke(0)<=0:
        xa:=0.5[rescale_from.left,rescale_from.right];
        xb:=0.5[rescale_from.left,rescale_from.right];
      else:
        xa:=infinity;
        xb:=-infinity;
        for i=1 upto sp-1:
          if obstacktype[i]=otstroke:
            if (xpart llcorner obstackp[i])<xa:
              xa:=xpart llcorner obstackp[i];
            fi;
            if (xpart lrcorner obstackp[i])>xb:
              xb:=xpart lrcorner obstackp[i];
            fi;
          fi;
        endfor;
      fi;
      % compute bearings and widths
      lf=xa-rescale_from.left;
      rf=rescale_from.right-xb;
      lf+rf+wf=rescale_from.right-rescale_from.left;
      lt+rt+wt=rescale_to.right-rescale_to.left;
      (lt,rt)=whatever[(0,0),(lf,rf)];
      wt=ypart (width_curve intersectionpoint ((wf,-infinity)--(wf,infinity)));
      % find transformation
      if wf>0:
        (rescale_from.left+lf,rescale_from.bottom) transformed t=
          (rescale_to.left+lt,rescale_to.bottom-rescale_skew);
        (rescale_from.left+lf,rescale_from.top) transformed t=
          (rescale_to.left+lt,rescale_to.top-rescale_skew);
        (rescale_from.right-rf,rescale_from.bottom) transformed t=
          (rescale_to.right-rt,rescale_to.bottom+rescale_skew);
      else:
        (rescale_from.left+lf,rescale_from.bottom) transformed t=
          (rescale_to.left+lt,rescale_to.bottom);
        (rescale_from.left+lf,rescale_from.top) transformed t=
          (rescale_to.left+lt,rescale_to.top);
        (rescale_from.left+lf+1,rescale_from.bottom) transformed t=
          (rescale_to.left+lt+1,rescale_to.bottom);
      fi;
      pair cp;
      transform st;
      cp:=((rescale_to.left+rescale_to.right)/2,rescale_to.bottom);
      cp transformed st=cp;
      cp+(100,0) transformed st=cp+(100,0);
      cp+(0,100) transformed st=cp+(rescale_slant/10,100);
      t:=t transformed st;
    endgroup; fi;
    t
  endgroup
enddef;

% solve the quadratic equation ax^2+bx+c=0, including pathological cases
vardef solve_quadratic(expr a,b,c) =
  if a=0:
    if b=0:
      if c=0:
        (0,whatever)
      else:
        (whatever,whatever)
      fi
    else:
      (-c/b,whatever)
    fi
  elseif abs(a)<abs(b)/500:
    (whatever,whatever)
  else:
    save d;
    numeric d;
    d=b*b-4*a*c;
    if d>0:
      ((-b-sqrt(d),-b+sqrt(d))/(2*a))
    elseif d=0:
      (-b/(2*a),whatever)
    else:
      (whatever,whatever)
    fi
  fi
enddef;

% find the t-values of any inflection points of subpath (0,1) of p
vardef segment_inflections(expr p) =
  save x,y,c;
  numeric x[],y[],c[];

  % normalize to prevent numerical misbehaviour
  z10=(point 0 of p)+z22;
  z11=(postcontrol 0 of p)+z22;
  z12=(precontrol 1 of p)+z22;
  z13=(point 1 of p)+z22;
  z10+z11+z12+z13=(0,0);
  c10=(abs(z10)+abs(z11)+abs(z12)+abs(z13))/4;
  if c10<0.1:
    c10:=0.1;
  fi;
  z0=z10/c10;
  z1=z11/c10;
  z2=z12/c10;
  z3=z13/c10;

  % abort if points are close enough to collinear
  if (abs(x0*y1-x1*y0)<0.01) and (abs(x2*y3-x3*y2)<0.01):
    (whatever,whatever)
  else:

    % find t-values at which |z’(t) cross z”(t)|=0
    c2=y0*( -1*x1 +2*x2 -x3)
      +y1*( x0 -3*x2 +2*x3)
      +y2*(-2*x0 +3*x1 -x3)
      +y3*( x0 -2*x1 +x2 );

    c1=y0*( 2*x1 -3*x2 +x3)
      +y1*(-2*x0 +3*x2 -x3)
      +y2*( 3*x0 -3*x1 )
      +y3*( -x0 +x1 );

    c0=y0*( -x1 +x2)
      +y1*( x0 -x2)
      +y2*( -x0 +x1 );

    z20=solve_quadratic(c2,c1,c0);

    % filter and sort to find points properly within the segment
    if known x20:
      if (x20>0.01) and (x20<0.99):
        x21=x20;
      fi;
    fi;
    if known y20:
      if (y20>0.01) and (y20<0.99):
        y21=y20;
      fi;
    fi;
    if known x21:
      z21
    else:
      (y21,x21)
    fi
  fi
enddef;

% version of insert_nodes modified to *always* insert, which is needed
% to keep node numbers in sync on pen-size-control paths
vardef really_insert_nodes(expr p)(text t) =
 save j_, p_, s_, t_; path p_; p_:=p;
 t_:=0;
 for i_:=t:
   t_[incr t_]=arclength(subpath(0,i_ mod length(p_)) of p_);
 endfor
 for i_:=1 upto t_:
  s_:=arctime t_[i_] of p_;
  p_:=(subpath (0, s_) of p_) && (subpath (s_,length p_) of p_)
    if cycle p_: & cycle fi;
 endfor;
 p_
enddef;

% render a single segment - pulled out to make it easier to override
def tsu_render_segment(expr i,p,q) =
  if do_alternation and obstackba.boalternate[i]:
    default_nib:=fix_nib(obstackna.bosize[i]*tsu_brush_max
                           *tsu_brush_shape,
                         obstackna.bosize[i]*tsu_brush_max
                           *tsu_brush_shape,
                         0);
  else:
    default_nib:=fix_nib(obstackna.bosize[i]*tsu_brush_max,
                         obstackna.bosize[i]*tsu_brush_max
```

```
                           *tsu_brush_shape,
                         tsu_brush_angle);
  fi;
  path mytip[],glyph;
  for l=0 step 1 until length(p):
    mytip[l]:=tsu_brush_tip(l,p,q,obstackna.bosize[i],s<1,
      t>(length obstackp[i])-1,obstackba.boalternate[i]);
  endfor;
  pen_stroke(for ell=0 step 1 until length(p):
      if sharp_corners and known obstacknaa.botip[i][ltime[ell]]:
        tip(obstacknaa.botip[i][ltime[ell]])(ell)
      else:
        tsu_brush_opt(mytip[ell])(ell)
      fi
    endfor)(p if abs((point infinity of p)-(point 0 of p))<1:
      --(point 0.001 of p)
    fi)(glyph);
  glstk[ngls]:=regenerate(glyph);
  ngls:=ngls+1;
  for l=0 step 1 until length(p):
    si:=floor (ltime[l]+0.5);
    if (abs(ltime[l]-si)<0.05) and known obstacknaa.boserif[i][si]:
      tsu_serif.choose(obstacknaa.boserif[i][si],
        point l of p,direction l of p,l,
        obstackna.bosize[i],tsu_brush_tip_size(l,q));
      write (”SERIF ”&(decimal obstacknaa.boserif[i][si])&” ”&
            (decimal xpart point l of p)&”,”&
            (decimal ypart point l of p)) to ”proof.prf”;
    fi;
  endfor;
enddef;

def tsu_render_in_circle(expr fitcircle) =
  %
  % find and apply rescaling xform
  %
  transform tsu_rescaling_xf;
  tsu_rescaling_xf:=tsu_rescale_xform;
  for i=1 upto sp-1:
    if known obstackp[i]:
      obstackp[i]:=obstackp[i] transformed tsu_rescaling_xf;
    fi;
    if known obstackt[i]:
      obstackt[i]:=obstackt[i] transformed tsu_rescaling_xf;
    fi;
  endfor;
  %
  % main render
```

```
%
for i=1 upto sp-1: if obstacktype[i]=othook:
  if obstackn[i]=hsmain_render:
    scantokens obstacks[i];
  fi;
fi; endfor;
begingroup
  numeric i,j,k,l,s,t,si,ngls,flati;
  path bqi,p,q,glstk[];
  ngls:=0;
  flati:=1;
  for i=1 upto sp-1: if obstacktype[i]=otstroke:
    if unknown obstackba.boalternate[i]:
      obstackba.boalternate[i]:=false;
    fi;
% message "suffix " & str i;
    bqi:=obstackq[i] transformed tsu_brush_xf;
    s:=0;
    for j=0 step 1 until (length obstackp[i])-1:
       k:=j+1;
% message " j=" & decimal j & " thr " & decimal (xpart point j of bqi)
% & "/" & decimal (xpart point k of bqi);
       if ((xpart point j of bqi)<1)
           and ((xpart point k of bqi)>=1):
% message " START";
         if (xpart point k of bqi)=1:
           s:=k;
         else:
           s:=j+(xpart ((subpath (j,k) of bqi)
             intersectiontimes ((1,-infinity)
             --(1,infinity))));
         fi;
       fi;
       if ((((xpart point j of bqi)>=1) and ((xpart point k of bqi)<1))
         or (k=length obstackp[i])):
% message " END";
         if (xpart point k of bqi)>=1:
           t:=k;
         else:
           t:=j+(xpart ((subpath (j,k) of bqi)
             intersectiontimes ((1,-infinity)
             --(1,infinity))));
         fi;
         if ((t-s)>0.02) and (obstackna.bosize[i]>0):
           boolean is_cycle;
           is_cycle:=((point s of obstackp[i])=(point t of obstackp[i]));
           p:=subpath (s,t) of obstackp[i];
           q:=subpath (s,t) of bqi;
```

```
numeric ltime[];
ltime[0]:=s;
for l=1 step 1 until (length p)-1:
  ltime[l]:=floor (s+l);
endfor;
ltime[length p]:=t;
l:=0;
forever:
  exitif l=length p;
  begingroup
    save x,y;
    numeric x[],y[];
    z10=segment_inflections(subpath (l,l+1) of p);
    if known x10:
      p:=really_insert_nodes(p)(l+x10);
      q:=really_insert_nodes(q)(l+x10);
      for ll=length p step -1 until l+1:
        ltime[ll]:=ltime[ll-1];
      endfor;
      ltime[l+1]:=x10[ltime[l],ltime[l+2]];
    else:
      z0=(point l of p)/100;
      z1=(postcontrol l of p)/100;
      z2=(precontrol (l+1) of p)/100;
      z3=(point (l+1) of p)/100;
      if if abs(z2-z1)>0.1: ((z1-z0) dotprod (z3-z2))
                         /((z2-z1) dotprod (z2-z1))
         else: 1 fi<0.5:
        p:=really_insert_nodes(p)(l+0.5);
        q:=really_insert_nodes(q)(l+0.5);
        for ll=length p step -1 until l+1:
          ltime[ll]:=ltime[ll-1];
        endfor;
        ltime[l+1]:=0.5[ltime[l],ltime[l+2]];
      else:
        l:=l+1;
      fi;
    fi;
  endgroup;
endfor;
write (”SEGMENT ”&(decimal flati)&” ”&(decimal s)&” ”&(decimal t))
  to ”proof.prf”;
for lcbj=0 upto length p:
  write (”POINT ”&(decimal flati)&” ”&(decimal ltime[lcbj])&” ”
    &(decimal xpart point lcbj of p)&” ”
    &(decimal ypart point lcbj of p)) to ”proof.prf”;
endfor;
flati:=flati+1;
```

```
         tsu_render_segment(i,p,q);
       fi;
     fi;
  endfor;
elseif obstacktype[i]=otlcblob:
  glstk[ngls]:=regenerate(obstackp[i]);
  ngls:=ngls+1;
fi; endfor;
%
% handle bounding circle
%
if xxpart fitcircle>0:
  begingroup
    save d,tmppt,pind,xpt,pts,pcnt,tmpxf;
    pair pts[];
    transform d;
    pcnt:=0;
    for j=0 upto ngls-1:
      for i=0 step 0.1 until length glstk[j]:
        pts[pcnt]:=point i of glstk[j];
        pcnt:=pcnt+1;
      endfor
    endfor;
    save lowpt; numeric lowpt;
    lowpt:=0;
    for i=0 upto pcnt-2:
      for j=i+1 upto pcnt-1:
        if (i>=lowpt) and (j>=lowpt) and (abs(pts[i]-pts[j])<2):
          swap_pts(j,lowpt);
          lowpt:=lowpt+1;
        fi;
      endfor;
    endfor;
    d:=bcircle.internal(lowpt,pcnt,pcnt);
    transform tmpxf;
    tmpxf=identity shifted (((0,0) transformed fitcircle)-
                           ((0,0) transformed d));
    for j=0 upto ngls-1:
      glstk[j]:=glstk[j] transformed tmpxf;
    endfor;
  endgroup
fi;
%
% finally render it all
%
for i=0 upto ngls-1:
  dangerousFill glstk[i];
endfor;
```

```
%
% write misc. proof file stuff
%
blobcount:=0;
boxcount:=0;
for i=1 upto sp-1:
  if obstacktype[i]=otlcblob:
    begingroup
      save spt,n;
      pair spt;
      spt:=(0,0);
      n:=0;
      for j=1 upto length obstackp[i]:
        n:=n+1;
        spt:=spt+(point j of obstackp[i]);
      endfor;
      spt:=spt/n;
      blobcount:=blobcount+1;
      write (”BLOBCENTRE ”&(decimal blobcount)&” ”
        &(decimal xpart spt)&” ”&(decimal ypart spt)) to ”proof.prf”;
    endgroup;
  elseif obstacktype[i]=otpbox:
    boxcount:=boxcount+1;
    write (”PBOX ”&
           (decimal boxcount)&” ”&
           (decimal xpart ((0,0) transformed obstackt[i]))&” ”&
           (decimal ypart ((0,0) transformed obstackt[i]))&” ”&
           (decimal xpart ((1,0) transformed obstackt[i]))&” ”&
           (decimal ypart ((1,0) transformed obstackt[i]))&” ”&
           (decimal xpart ((1,1) transformed obstackt[i]))&” ”&
           (decimal ypart ((1,1) transformed obstackt[i]))&” ”&
           (decimal xpart ((0,1) transformed obstackt[i]))&” ”&
           (decimal ypart ((0,1) transformed obstackt[i]))&” ”’&
           obstacks[i]&””) to ”proof.prf”;
    if known obstackba.botoexpand[i]:
      if obstackba.botoexpand[i]:
        errmessage ”Unexpanded PBOX: ” & obstacks[i];
      fi;
    fi;
  elseif obstacktype[i]=otanchor:
    begingroup
      save topanchor;
      numeric topanchor;
      topanchor:=i;
      for j:=sp-1 downto i+1:
        if obstacktype[j]=otanchor:
          if obstackn[j]=obstackn[i]:
            topanchor:=j;
```

```
              fi;
            fi;
            exitif topanchor<>i;
          endfor;
          if topanchor=i:
            write ("ANCHOR "&
                   (decimal obstackn[i])&" "&
                   (decimal xpart ((-35,0) transformed obstackt[i]))&" "&
                   (decimal ypart ((-35,0) transformed obstackt[i]))&" "&
                   (decimal xpart ((35,0) transformed obstackt[i]))&" "&
                   (decimal ypart ((35,0) transformed obstackt[i]))&" "&
                   (decimal xpart ((0,-35) transformed obstackt[i]))&" "&
                   (decimal ypart ((0,-35) transformed obstackt[i]))&" "&
                   (decimal xpart ((0,35) transformed obstackt[i]))&" "&
                   (decimal ypart ((0,35) transformed obstackt[i])))
                    to "proof.prf";
          fi;
        endgroup;
      fi;
    endfor;
  endgroup;
enddef;

% the usual case - just render it without fitting into a circle
def tsu_render =
  tsu_render_in_circle(identity scaled -1);
enddef;

transform tsu_xf.smallkana;

tsu_xf.smallkana = identity shifted (-500,0) scaled 5.5/8 shifted (500,0);

def tsu_xform(expr xform)(text curves) =
  begingroup
    save txfsp,zc,zs;
    txfsp:=sp;
    curves;
    numeric zs;
    zs:=abs(((0,0) transformed xform)
            -((1,0) transformed xform))
       *abs(((0,0) transformed xform)
           -((0,1) transformed xform));
    size_scale:=zs**0.16667;
    for i=txfsp upto sp-1:
      if known obstackp[i]:
        if known obstackba.bokeepshape[i]:
          if obstackba.bokeepshape[i]:
            pair zc;
            zc:=0.5[llcorner obstackp[i],urcorner obstackp[i]];
            obstackp[i]:=obstackp[i] shifted (-zc) scaled (yypart xform)
              shifted (zc transformed xform);
          else:
            obstackp[i]:=obstackp[i] transformed xform;
          fi;
        else:
          obstackp[i]:=obstackp[i] transformed xform;
        fi;
      fi;
      if known obstackna.bosize[i]:
        obstackna.bosize[i]:=obstackna.bosize[i]*size_scale;
      fi;
      if known obstackt[i]:
        if (xxpart obstackt[i]=1) and (yypart obstackt[i]=1)
          and (xypart obstackt[i]=0) and (yxpart obstackt[i]=0):
          obstackt[i]:=obstackt[i]
            shifted (((0,0) transformed obstackt[i] transformed xform)-
                     ((0,0) transformed obstackt[i]));
        else:
          obstackt[i]:=obstackt[i] transformed xform;
        fi;
      fi;
    endfor;
  endgroup;
enddef;



def anc_upper = 1 enddef;
def anc_grave = 2 enddef;
def anc_acute = 3 enddef;
def anc_wide = 4 enddef;
def anc_tilde = 5 enddef;
def anc_ring = 6 enddef;
def anc_caron_comma = 7 enddef;
def anc_dakuten = 8 enddef;
def anc_lower = 9 enddef;
def anc_lower_connect = 10 enddef;
def anc_centre = 11 enddef;
def anc_iching_line(expr lnum) = (11+lnum) enddef;

transform accent_default[];
boolean accent_has_default[];
max_accent_seen:=0;

def tsu_default_anchor(expr aindex,avalue) =
  if numeric avalue:
    write (”DEFAULTANCHOR ”&(decimal aindex)&” FALSE”) to ”proof.prf”;
    accent_has_default[aindex]:=false;
  elseif transform avalue:
    write (”DEFAULTANCHOR ”&
           (decimal aindex)&” ”&
           (decimal xpart ((0,0) transformed avalue
                          transformed tsu_rescale_xform))&” ”&
           (decimal ypart ((0,0) transformed avalue
                          transformed tsu_rescale_xform)))
            to ”proof.prf”;
    accent_has_default[aindex]:=true;
  elseif pair avalue:
    write (”DEFAULTANCHOR ”&
           (decimal aindex)&” ”&
           (decimal xpart (avalue transformed tsu_rescale_xform))&” ”&
           (decimal ypart (avalue transformed tsu_rescale_xform))
            to ”proof.prf”;
    accent_has_default[aindex]:=true;
  fi;
  if aindex>max_accent_seen:
    max_accent_seen:=aindex;
  fi;
enddef;

```

```

% figure out size of brush
vardef calc_mbrush_size =
  numeric mbrush_width,mbrush_height,alternate_adjust;
  (mbrush_width,mbrush_height)=urcorner (
     fullcircle xscaled (tsu_brush_max*100)
     yscaled (tsu_brush_max*tsu_brush_shape*100)
     rotated tsu_brush_angle
  );
  alternate_adjust:=abs(mbrush_height-mbrush_width);
  if tsu_brush_max>0.75:
    serif_size:=2;
  else:
    serif_size:=2*((tsu_brush_max/0.75)**(1/3));
  fi;
  mincho_blob_size:=sqrt(tsu_brush_max/0.75);
enddef;

calc_mbrush_size;
```

fntbase.mp

```
%
% Tsukurimashou ”font base” macros
%
% THIS FILE IS PUBLIC DOMAIN NOTWITHSTANDING THE COPYRIGHT ON THE
% OVERALL TSUKURIMASHOU PACKAGE
%
% This file is based on the files ”fontbase.mp” and ”plain_ex.mp” from the
% METATYPE1 package version 0.55. Those files contain no copyright-related
% notices of their own, but the README for METATYPE1 version 0.55 contains
% the following notices (in English and Polish; the slashes are verbatim
% from the original and presumably are some convention for expressing
% non-ASCII Polish letters in the ASCII file):
%
% This is METATYPE1 package - a tool for creating Type 1 fonts using
% METAPOST. The package belongs to public domain (no copyrights,
% copylefts, copyups, copydowns, etc.).
% Version: 0.55 (16.09.2009; a tentative version, released along with
% the sources of the Latin Modern fonts ver. 2.003)
% Author: JNS team <JNSteam@gust.org.pl>
%
% To jest pakiet METATYPE1 - narz/edzie do tworzenia font/ow Type 1
% za pomoc/a systemu METAPOST. Pakiet stanowi dobro wsp/olne
% (/zadnych copyright/ow, copyleft/ow, copyup/ow, copydown/ow, etc.).
% Wersja: 0.55 (16.09.2009 - wersja opublikowana wraz z wersj/a
% /xr/od/low/a 2.003 pakietu font/ow Latin Modern)
% Autorstwo: JNS team <JNSteam@gust.org.pl>
%
% Although I assert my general right to claim copyright on work of my own
% that draws from public domain source materials, I nonetheless am releasing
% this file to the public domain in an effort to maintain the spirit of the
% JNS team's release above.
%
% This program is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
%
% Matthew Skala
% mskala@ansuz.sooke.bc.ca
%

```

General Library Functions

```
% GENERAL LIBRARY FUNCTIONS

% inclusion lock written explicitly so as not to depend on preintro.mp
if known already_included.fntbase:
  endinput;
fi;
boolean already_included.fntbase;
already_included.fntbase:=true;

% gobble a text argument
def killtext text t = enddef; % absent from older versions of plain.mf

% Knuthian tradition unit definitions
mm#=2.84528; pt#=1; dd#=1.07001; bp#=1.00375; cm#=28.45276; pc#=12;
cc#=12.84010; in#=72.27;

% numeric functions
vardef tand primary a = sind(a)/cosd(a) enddef;
vardef cotd primary a = cosd(a)/sind(a) enddef;
vardef signum primary x = if x>0: 1 elseif x<0: -1 else: 0 fi enddef;
primarydef w dotnorm z =
 begingroup
  save w_, z_, lw_, lz_; pair w_, z_;
  lw_=abs(w); w_:=w if lw_>0: /lw_ fi;
  lz_=abs(z); z_:=z if lz_>0: /lz_ fi;
  (xpart w_ * xpart z_ + ypart w_ * ypart z_)
 endgroup
enddef;

% expand ”decimal” to cover some other data types
let ori_decimal=decimal;
def decimal primary n =
 (
  if path n:
   for i_=0 upto length(n)-1: if i_>0: & ” ” & fi
    decimal(point i_ of n) & ” ” & decimal(postcontrol i_ of n) & ” ” &
     decimal(precontrol i_+1 of n) & ” ” & decimal(point i_+1 of n)
   endfor
  elseif color n: ori_decimal(redpart(n)) & ” ” &
   ori_decimal(greenpart(n)) & ” ” & ori_decimal(bluepart(n))
  elseif pair n: ori_decimal(xpart(n)) & ” ” & ori_decimal(ypart(n))
  else: ori_decimal(n) fi
 )
enddef;

% The definition of |postdir| and |predir| given below is
% based on the following observation, being the consequence
% of l’H\`opital’s rule: consider a B\’ezier segment
% |a .. controls b and c .. d|; normally, the vector $\vec{ab}$
% determines the “post” direction at node $a$; if $b$
% coincides with $a$, then the vector $\vec{ac}$ determines
% the direction; if also $c$ coincides coincides with $a$,
% then the last resort is the vector $\vec{ad}$; if even $d$
% coincides with $a$, the B\'ezier segment is degenerated,
% and can be removed (a similar argumentation can be provided
% for the “pre” direction at node $d$).

% Previous, insufficiently robust definitions:
% |vardef predir expr t of p = (point t of p)-(precontrol t of p) enddef;|
% |vardef postdir expr t of p = (postcontrol t of p)-(point t of p) enddef;|
% |vardef udir expr t of p = unitvector(direction t of p) enddef;|

% New, more general definitions:
vardef gendir expr t of p =
 predir t of p + postdir t of p % |direction|-compatible definition
enddef;
vardef predir expr t of p =
 save a_,b_,c_,d_,s_,t_; pair a_,b_,c_,d_; path s_; t_:=t;
 if not cycle p: if t<0: t_:=0; elseif t>length(p): t_:=length(p); fi fi
  s_=subpath (ceiling t_-1,t_) of p;
  a_=point 0 of s_;
  b_=postcontrol 0 of s_; % |b_<>postcontrol t-1 of p| for |t=0|
  c_=precontrol 1 of s_;
  d_=point 1 of s_;
  if d_<>c_: d_-c_
  elseif d_<>b_: d_-b_
  elseif d_<>a_: d_-a_
  else: (0,0)
 fi
enddef;

vardef postdir expr t of p =
 save a_,b_,c_,d_,s_,t_; pair a_,b_,c_,d_; path s_; t_:=t;
 if not cycle p: if t<0: t_:=0; elseif t>length(p): t_:=length(p); fi fi
  s_=subpath (t_,floor t_+1) of p;
  a_=point 0 of s_;
  b_=postcontrol 0 of s_;
  c_=precontrol 1 of s_; % |c_<>precontrol t+1 of p| for |t=length p|
  d_=point 1 of s_;
  if a_<>b_: b_-a_
  elseif a_<>c_: c_-a_
  elseif a_<>d_: d_-a_
  else: (0,0)
 fi
enddef;

% Definitions related to “pre-” and “post-”
vardef udir expr t of p = unitvector(gendir t of p) enddef;
vardef upredir expr t of p = unitvector(predir t of p) enddef;
vardef upostdir expr t of p = unitvector(postdir t of p) enddef;
vardef pos_subpath expr z of p =
 if not cycle p: subpath z of p else:
  if xpart(z)<=ypart(z): subpath z of p
  else: subpath (xpart(z),ypart(z)+length(p)) of p fi
 fi
enddef;

vardef posttension expr t of p = % “The \MF{}book”, ex. 14.15
 save q__; path q__;
 q__=point t of p {direction t of p} .. {direction t+1 of p} point t+1 of p;
 length(postcontrol 0 of q__ - point 0 of q__)/
 length(postcontrol t of p - point t of p)% doesn’t work for “straight lines”
enddef;
vardef pretension expr t of p = % ditto
 save q__; path q__;
 q__=point t-1 of p {direction t-1 of p} .. {direction t of p} point t of p;
 length(precontrol 1 of q__ - point 1 of q__)/
 length(precontrol t of p - point t of p)% doesn’t work for “straight lines”
enddef;

% The two macros below, |path_eq| and |inside| macros, might have been
% primitives. The macro |path_eq| is obvious; |a inside b| returns true
% if the bounding box of |a| is inside the bounding box of |b|, which
% may be misleading; think, for example of:
% |fullcircle inside unitsquare shifted (-1/2,-1/2) scaled .9 rotated 45|.
% For most curves occuring in fonts, however, one can safely infere
% that if |a inside b| holds, then |a| is inside |b|.
vardef path_eq(expr a,b)=
 save i__,l__,r__; boolean r__;
 r__:=(length(a)=length(b)) and (cycle a= cycle b);
 if r__:
  i__:=0; l__:=length(a) if cycle a: -1 fi;
  forever:
   r__:=(point i__ of a = point i__ of b); exitif not r__;
   r__:=(precontrol i__ of a = precontrol i__ of b); exitif not r__;
   r__:=(postcontrol i__ of a = postcontrol i__ of b); exitif not r__;
   exitif incr i__>l__;
 endfor fi
 r__
enddef;

tertiarydef a inside b =
 if path a: % |and path b|
  (xpart llcorner b < xpart llcorner a) and
  (xpart urcorner b > xpart urcorner a) and
  (ypart llcorner b < ypart llcorner a) and
  (ypart urcorner b > ypart urcorner a)
 else: % |numeric a and pair b|
  (a>=xpart b) and (a<=ypart b)
 fi
enddef;

% The macro |&&| is to be used instead of the |&| operator if the respective
% ends of paths coincide only approximately; using |..| instead would add
% unwanted tiny B\'ezier segments. The macro is somewhat “left-handed,”
% i.e., it does not consider the expression that follow the macro, therefore,
% it can be used before the ‘|cycle|’ command; if the argument |p| of the
% macro |amp_amp_| is a |pair|, it is just ignored which may be
% considered hardly intuitive.
def && = amp_amp_ whatever enddef;
tertiarydef p amp_amp_ q =
 if not pair p:
  (subpath(0,length(p)-1) of p) .. controls (postcontrol length(p)-1 of p)
   and (precontrol length(p) of p) ..
 fi
enddef;

vardef extrapolate expr t of b = % |t| pair, |b| B\'ezier segment
 clearxy;
 Casteljau(xpart(t)) = point 0 of b;
 Casteljau(1/3[xpart(t),ypart(t)]) = point 1/3 of b;
 Casteljau(2/3[xpart(t),ypart(t)]) = point 2/3 of b;
 Casteljau(ypart(t)) = point 1 of b;
 z0 .. controls z1 and z2 .. z3
enddef;

def Casteljau(expr t) =
 t[t[t[z0,z1], t[z1,z2] ], t[t[z1,z2], t[z2,z3] ] ]
enddef;

vardef elongation_to_times(expr ea,eb) =
 % negative parameter values are admissible; they are meant for |pen_stroke|
 (if ea<0: - fi 1/(abs(ea)+1), eb/(abs(eb)+1))
enddef;

% A numerical function ‘|point_line_dist|’ takes as a parameter
% three |pair| expressions and returns a (signed) value of the distance
% of the first parameter from the line defined by the other two.
% It is referred to in the ‘|is_line|’ function.

vardef point_line_dist(expr a,b,c) =
 clearxy; save d_; d_=sqrt(length(b-c));
 z0=a/d_; z1=b/d_; z2=c/d_;
 (x2-x1)*(y1-y0)-(x1-x0)*(y2-y1)
```

```
enddef;

% The idea of calculation of a turning angle
% between two vectors, employed in the definition of the function
% ‘turn_ang,’ is based on the following observation:
vardef turn_ang(expr za,zb) =
 if (abs(za)>=1/1000) and (abs(zb)>=1/1000): % |eps| may be not enough
  angle(unitvector(za) zscaled (unitvector(zb) reflectedabout (origin,right)))
 else: whatever fi
enddef;

% A Boolean function ‘|is_line|’ checks whether a given B\'ezier segment
% is a straight line. For large segments (fonts) it makes sense to specify
% a numerical parameter |is_line_off>=0|; it defines a maximal acceptable
% distance of the control points of a B\'ezier arc from its secant
% (which corresponds to the distance between the arc and the secant
% circa |3/4is_line_off| for a symmetric, inflexionless arcs).
vardef is_line(expr B) =
 save r_; boolean r_;
 if known is_line_off:
  save a_;
  a_=length((point 1 of B)-(point 0 of B));
  r_=(-a_+arclength(B))<=(a_/infinity);
  if r_:
   r_:=(is_line_off>=abs(point_line_dist(
         postcontrol 0 of B, point 0 of B, point 1 of B))) and
       (is_line_off>=abs(point_line_dist(
         precontrol 1 of B, point 0 of B, point 1 of B)));
  fi
 else: % backward compatibility
  save a_,b_,c_,d_;
  a_=length((point 1 of B)-(point 0 of B));
  b_=length((postcontrol 0 of B)-(point 0 of B));
  c_=length((precontrol 1 of B)-(postcontrol 0 of B));
  d_=length((point 1 of B)-(precontrol 1 of B));
  r_=(-a_+b_+c_+d_ <= a_/infinity);
 fi
 r_
enddef;

% Abbreviations for a few simple yet useful phrases
def xyscaled primary p = xscaled xpart(p) yscaled ypart(p) enddef;
def yxscaled primary p = yscaled xpart(p) xscaled ypart(p) enddef;

% The macro |insert_nodes| inserts additional nodes at given non-integer
% non-repeating times |t| into a given path |p|.
% The code would be a bit longer without ‘|arclength|’ and ‘|arctime|.’
% The macro can be useful in some cases in the context of finding
% the envelopes of pen-stroked paths (avoiding inflection
% points—see below).
vardef insert_nodes(expr p)(text t) =
 save j_, p_, s_, t_; path p_; p_:=p;
 t_:=0;
 for i_:=t:
  if round(i_)<>i_: % ignore integer times
   t_[incr t_]=arclength(subpath(0,i_ mod length(p_)) of p_);
  fi
 endfor
 for i_:=1 upto t_:
  s_:=arctime t_[i_] of p_;
  if abs(round(s_)-s_)>eps: % ignore repeating times; is |eps| OK?
   p_:=(subpath (0, s_) of p_) && (subpath (s_,length p_) of p_)
    if cycle p_: & cycle fi;
  fi
 endfor;
 p_
enddef;

% get rid of degeneracies
def regenerate(expr p) =
  if (xpart urcorner p-xpart ulcorner p<5)
    and (ypart ulcorner p-ypart llcorner p<5):
    p
  else:
    begingroup
      save q;
      path q;
      for t=1 step 1 until length p:
        if abs((point t of p)-(point (t-1) of p))>3:
          if unknown q:
            q:=subpath (t-1,t) of p;
          elseif length(q)=1:
            q:=(point 0 of q)..
                controls (postcontrol 0 of q) and (precontrol 1 of q)..
              (0.5[point 1 of q,point t-1 of p])..
                controls (postcontrol t-1 of p) and (precontrol t of p)..
              (point t of p);
          else:
            q:=(subpath (0,length(q)-1) of q)..
                controls (postcontrol length(q)-1 of q)
                  and (precontrol length(q) of q)..
              (0.5[point length(q) of q,point t-1 of p])..
                controls (postcontrol t-1 of p) and (precontrol t of p)..
              (point t of p);
          fi;
        fi;
      endfor;
      if cycle p:
        q:=subpath (0,length(q)-1) of q..
             controls (postcontrol length(q)-1 of q)
               and (precontrol length(q) of q)..
           cycle;
      fi;
      q
    endgroup
  fi
enddef;

% like Fill, but doesn't complain about turning number
def dangerousFill text glist =
 begingroup
  save h_; path h_;
  for g_:=glist:
   h_:=g_ start.default; % JMN's suggestion
   if glyph_usage div store = 1: % storing
    glyph_stored.glyph_name[incr glyph_stored.glyph_name.num]=h_;
   fi
   glyph_list[incr glyph_list.num]:=round_node_values(h_ italicized);
   update_glyph_bb(glyph_list[glyph_list.num]);
  endfor;
 endgroup
enddef;
```

Prefix And Suffix Handling

```
% PREFIX AND SUFFIX HANDLING

% A method, entangled a bit and not particularly robust, of testing whether
% a parameter is a {\it string\/} expression or a {\it suffix}.
% (Remark: |is_suffix((a))| or |is_suffix(a+b)| returns |true|;
% |is_suffix(((a)))| causes \MP{} to report an error).
vardef is_suffix(text suffix_or_not_suffix) =
 save the_suffix_; string the_suffix_; is_suffix_ suffix_or_not_suffix;
 the_suffix_<>""
enddef;
def is_suffix_ suffix $ = the_suffix_:= str $; killtext enddef;

% suffix munging

def store_prec_obj = store_prec_obj_ whatever enddef;
primarydef a store_prec_obj_ b = hide(def prec_obj = a enddef) enddef;
```

```

% primarydef a sub b =
% if path a: (pos_subpath b of a) elseif string a: (substring b of a) fi
% enddef;

def node = store_prec_obj node_ enddef;
vardef node_@# primary a =
 if str @#=”x”: xpart(point a of prec_obj)
 elseif str @#=”y”: ypart(point a of prec_obj)
 elseif str @#=””: point a of prec_obj
 else:
  errhelp ”The operator ‘node’ works only with ‘x’, ‘y’ or an empty suffixes.”;
  errmessage ”PX: improper usage of ‘node’”;
 fi
enddef;

def first suffix $ =
 if str $=”at”: % moves the first point of a path to a specified location
  store_prec_obj prec_obj shifted -(point 0 of prec_obj) shifted
 else: node$(0) fi
enddef;
def last suffix $ =
 if str $=”at”: % moves the last point of a path to a specified location
  store_prec_obj prec_obj shifted
   -(point if cycle prec_obj: 0 else: infinity fi of prec_obj) shifted
 else: node$(if cycle prec_obj: 0 else: infinity fi) fi
enddef;

% Neat macros excerpted from John D. Hobby’s boxes.mp macro package

% Find the length of the prefix of string |s| for which |cond| is true for
% each character |c| of the prefix
vardef genericize_prefix(expr s)(text cond) =
 save i_, c_; string c_;
 i_ = 0;
 forever:
  c_ := substring (i_,i_+1) of s;
  exitunless cond; exitif incr i_=length s;
 endfor
 i_
enddef;

% Take a string returned by the |str| operator and return the same string
% with explicit numeric subscripts replaced by generic subscript symbols []:
vardef genericize(expr s) =
 save res_, s_, l_; string res_, s_;
 res_=””; % result so far
 s_ =s; % left to process
 forever: exitif s_=””;
  l_:=genericize_prefix(s_, (c_<>”[”) and ((c_<”0”) or (c_>”9”)));
  res_:=res_ & substring (0,l_) of s_;
  s_:=substring (l_,infinity) of s_;
  if s_<>””:
   res_ := res_ & ”[]”;
   l_ :=if s_>=”[”: 1+genericize_prefix(s_, c_<>”]”)
    else: genericize_prefix(s_, (c_=”.”) or (”0”<=c_) and (c_<=”9”)) fi;
   s_:=substring(l_,infinity) of s_;
  fi
 endfor
 res_
enddef;
```

A Module That Finds An Envelope Of A Path Drawn With A Pen

```
% A MODULE THAT FINDS AN ENVELOPE OF A PATH DRAWN WITH A PEN

% The following macros approximate the envelope of an elliptical or a razor
% pen. The exact solution is impossible—in general, the envelope is not
% a B\'ezier curve, therefore some heuristics is, in general, unavoidable.
% We assumed that the backbone of a figure is such that
% the envelope does not form loops at smoothly joined nodes. Moreover,
% all B\'ezier segments appearing in the process {\bf should not}
% contain inflection points (the reason for this limitation is the
% method of finding an approximation of a pen envelope). If the latter
% condition is not fulfilled, one may expect weird results (see the usage
% of the |...| operator in the code of |pen_stroke_edge|).

% We assume that slanting should not distort a pen. Therefore, if
% a glyph is to be slanted {\it after\/} expanding a stroke, which
% usually is the case, the envelope should be constructed with
% an {\it unslanted pen}. Macros |slant_stroke|, |unslant_stroke|,
% and |unslant_angle| are devised to facilitate handling this
% situation. These macros refer to the variable |slant_stroke_val|;
% it should be assigned a definite value prior to expanding stroke.
def slant_stroke =
 if known slant_stroke_val: slanted slant_stroke_val fi
enddef;
def unslant_stroke =
 if known slant_stroke_val: slanted -slant_stroke_val fi
enddef;
vardef unslant_angle(expr a) = angle(dir(a) unslant_stroke) enddef;
```

```
% Macro |fix_nib| returns a path. If |y_diam| parameter
% is 0, a “razor” pen (a segment) is returned, otherwise it is
% an approximation of an ellipse. We do our best to avoid unnecessary
% nodes, hence the approximation is somewhat complicated; another reason
% for the complication is that interpolation and affine transformations
% do not commute, therefore the appropriate nodes are found for
% the untransformed pen, and only then the pen is transformed.
% {\it Note\/}: So far, there is no explicit relation between a built-in
% \MP{} pen mechanism and the |fix_nib| operation, in particular,
% |beginfig| does not alter the setting of |default_nib|. Needs rethinking.

vardef fix_nib(expr x_diam, y_diam, rot_angle) =
 if (x_diam<>0) and (y_diam<>0): fix_elliptic_nib(x_diam, y_diam, rot_angle)
 elseif (x_diam<>0) and (y_diam=0): fix_razor_nib(x_diam, rot_angle)
 elseif (x_diam=0) and (y_diam<>0): fix_razor_nib(y_diam, rot_angle+90)
 else:
  errhelp ”I’ll use the default pen, but I’d suggest to cancel the job.”;
  errmessage ”PX: the null pen is not alowed”;
  default_nib
 fi
enddef;

vardef fix_razor_nib(expr x_diam, rot_angle) =
 ((-1/2x_diam,0)–(1/2x_diam,0)) rotated rot_angle unslant_stroke
enddef;

vardef fix_elliptic_nib(expr x_diam, y_diam, rot_angle) =
 save p_; path p_;
 % construct a temporary ellipse:
 p_:=fullcircle
  xscaled x_diam yscaled y_diam rotated rot_angle unslant_stroke;
 % construct an elliptic pen path having
 % 4 or, if necessary (heuristic), 6 nodes:
 (for d=up unslant_stroke, left,
  if (y_diam/x_diam<1/2) and (abs(rot_angle mod 90)>5):
    left rotated rot_angle unslant_stroke,
  fi
  down unslant_stroke, right,
  if (y_diam/x_diam<1/2) and (abs(rot_angle mod 90)>5):
    right rotated rot_angle unslant_stroke
  fi:
  (point(directiontime d of p_) of fullcircle)
   {direction (directiontime d of p_) of fullcircle}...
 endfor cycle) xscaled x_diam yscaled y_diam rotated rot_angle unslant_stroke
enddef;

% Arcs of a pen shorter than |ignore_nib_limit| will be joined together
% to form larger ones. Remember to adjust the parameter |ignore_nib_limit|
% if the size of |default_nib| is significantly changed.
newinternal ignore_nib_limit; ignore_nib_limit:=1;

path default_nib;
default_nib:=fix_nib(50,50,0); % hundred times as large as a default plain pen

newinternal default_elongation, default_join, default_cap;
default_elongation:=1/2;
default_join:=1;
  % 0 - tip, default elongation used
  % 1 - pen join, default elongation ignored
  % 2 - tip, default elongation ignored, elongation=0 used
default_cap:=1;
  % 0 - cut 90 rel
  % 1 - pen end

% |tangent_point|, |pen_join|, |pen_stroke_edge_|, and |pen_stroke_edge|
% are auxiliary macros, exploited by the main macro, i.e., |pen_stroke|.
vardef tangent_point(expr d,nib) = % |d| - direction of pen movement
 save a_;
 point if cycle nib: (directiontime d of nib) else:
  hide (a_:=turn_ang(d,(point 1 of nib)-(point 0 of nib)))
  if abs(a_ mod 180)<.1: 1/2 % emergency
  elseif a_<0: 0 else: 1 fi
 fi of nib
enddef;

vardef pen_join(expr a,b,c,nib)=
 % deleting superfluous nodes is based on the |arclength| operation
 % which, obviously, is not preserved after slanting, but let's hope
 % it does not matter (too much)
 save t_, m_, m__, ta_, tb_, p_; path p_;
 m_:=infinity; % will be the minimal length of |nib|'s segment
 for t_:=0 upto 1/2length(nib)-1:
  m__:=arclength(subpath(t_,t_+1) of nib);
  if m__<m_: m_:=m__; fi
 endfor
 if m_<ignore_nib_limit:
  message ”PX: the shortest nib segment < ignore_nib_limit (” &
    decimal(m__) & ” < ” & decimal(ignore_nib_limit) & ”)”;
 fi
 p_=nib shifted c;
 if cycle nib:
  ta_=directiontime a of p_; tb_=directiontime b of p_;
  p_:=pos_subpath(ta_,tb_) of p_;
  if arclength(p_)>ignore_nib_limit:
   for i_:=0,0:
    p_:=reverse p_; % short segments may appear at both ends
    if length(p_)>1: % optimization
     if arclength(subpath (0,1) of p_)<1/4ignore_nib_limit:
      % cf. the comment concerning |1/4ignore_nib_limit| in
      % |pen_stroke_edge| below
      p_:=(point 0 of p_) .. controls (postcontrol 1 of p_) and
        (precontrol 2 of p_) .. subpath (2,infinity) of p_;
    fi
   fi
   endfor
  else:
   p_:=(point 0 of p_){a}...{b}(point length(p_) of p_);
  fi
 else: % razor nib
  p_:=tangent_point(a,p_)--tangent_point(b,p_);
 fi
 p_
enddef;

% The finding of a pen envelope for a given B\'ezier segment,
% defined by nodes |a|, |b|, |c|, and |d|, begins with
% the placing the pen at the ends of the B\'ezier segment
% (i.e., at the points |a|, |d|) and finding the corresponding points
% |a'| and |d'| where the pen outline is parrallel to the direction
% of the original path at these points. Then, the outline is constructed.
% For |pen_stroke_method=0| (default), the envelope segment is constructed
% by setting the beginning and final directions (optionally, the direction
% at a given node can be ignored); for |pen_stroke_method=1| or 2
% an alternative (more elaborate) procedure is involved which explicitly
% computes control nodes |b'| and |c'| of the resulting path basing on
% a heuristic assumption that
% |length(b'-a')/length(b-a)| $\approx$
% |length(c'-d')/length(c-d)| $\approx$
% |length(a'-d')/length(a-d)|.\break
% The default method never produce concave edges because the operator |...|
% is used always; the alternative methods employs the operator
% |force_convex_edge| instead; for |pen_stroke_method=1| the convex edges
% are forced (i.e, inflexion points are being removed),
% for |pen_stroke_method=2| no forcing of convex edges takes place.
vardef extrapoline expr t of B = % the result may be not a single segment
 save l_, t_;
 (t_.a,t_.b)=t; % |0<=ta_<tb_<=1|!
 l_=arclength(B)/(t_.b-t_.a); l_.a=l_*t_.a; l_.b=l_*(1-t_.b);
 if t_.a>0: ((point 0 of B) - l_.a*(upostdir 0 of B))-- fi
  B
 if t_.b<1: -- ((point 1 of B) + l_.b*(upredir 1 of B)) fi
enddef;

vardef force_convex_edge(expr za, zb, zc, zd) =
 save a_, b_, c_, d_, z_;
 a_:=length(zd-za); b_:=length(zb-za); c_:=length(zc-zb); d_:=length(zd-zc);
 if (-a_+b_+c_+d_ > a_/infinity):
  if pen_stroke_method=2:
   za .. controls zb and zc .. zd
  else:
   if (a_>0.01) and (b_>0.01) and (c_>0.01) and (d_>0.01): % no degeneration...
    a_:=signum((za-zd) rotated -90 dotnorm (zb-za));
    b_:=signum((zb-za) rotated -90 dotnorm (zc-zb));
    c_:=signum((zc-zb) rotated -90 dotnorm (zd-zc));
    d_:=signum((zd-zc) rotated -90 dotnorm (za-zd));
    if ((a_<>b_) or (b_<>c_)) and (a_=d_):
     numeric b_, c_; pair z_;
     z_=b_[za,zb]=c_[zd,zc];
     za .. controls
      if b_<1: z_ else: zb fi and if c_<1: z_ else: zc fi
     .. zd
    else:
     za .. controls zb and zc .. zd
    fi
   else:
    za .. controls zb and zc .. zd
   fi
  fi
 else:
  za — zd
 fi
enddef;

vardef pen_stroke_edge_(expr b,b_nib,e_nib) = % |b| — B\'ezier segment
 save pa_,pb_,qa_,qb_,ra_,rb_,sa_,sb_;
 pair pa_,pb_,qa_,qb_,ra_,rb_,sa_,sb_;
 pa_=point 0 of b; ra_=(postcontrol 0 of b)-pa_; sa_=postdir 0 of b;
 pb_=point 1 of b; rb_=(precontrol 1 of b)-pb_; sb_=predir 1 of b;
 qa_=pa_ + tangent_point(sa_, b_nib);
 qb_=pb_ + tangent_point(sb_, e_nib);
 if pen_stroke_method=0:
  qa_ {sa_} ... {sb_} qb_
 elseif (pen_stroke_method=1) or (pen_stroke_method=2):
  save lp_,lq_; lp_=length(pb_-pa_); lq_=length(qb_-qa_);
  if 2lp_<lq_: % heuresis — too close nodes
   qa_ {sa_} ... {sb_} qb_
  else:
   force_convex_edge(qa_, qa_+lq_/lp_*ra_, qb_+lq_/lp_*rb_, qb_)
  fi
 else:
  errhelp ”Only the values 0,1 and 2 for ‘pen_stroke_method’ are admissible. ” &
   ”Better stop now.”;
  errmessage "PX: unknown pen stroke method (" &
   decimal(pen_stroke_method) & ")";
 fi
enddef;

vardef pen_stroke_edge@#(expr p) =
 save e_,l_,i_,i__; path e_[\\];
 l_:=length(p);
 for i_:=0 upto l_-1:
  e_[i_]=pen_stroke_edge_(subpath (i_,i_+1) of p,
   % |local_nib_@#(i_),local_nib_@#(i_+1));| % a nasty bug removed 20.08.2009
   local_nib_@#(i_),local_nib_@#((i_+1) if cycle p: mod l_ fi));
 endfor
 for i_:=0 upto l_ if cycle p: -1 else: -2 fi:
  i__:=(i_+1) mod l_;
  save t_;
  t_:=turn_ang(predir 1 of e_[i_], postdir 0 of e_[i__]);
  if if known t_: abs(t_)>1 else: false fi:
   save t_; (t_.a,t_.b)=e_[i_] intersectiontimes e_[i__];
   if t_.a>0:
    e_[i_]:=subpath (0,t_.a) of e_[i_];
    e_[i__]:=subpath (t_.b,1) of e_[i__];
   elseif known local_tip_@#(i__):
    save tx_, ty_, b_, b__, ei_, ei__; path ei_, ei__, ei_[], ei__[];
    (tx_,ty_)=local_tip_@#(i__);
    ei_:=if is_line(e_[i_]):
     (point 0 of e_[i_])-
     (1/abs(tx_))[point 0 of e_[i_], point 1 of e_[i_] ]
     elseif tx_<0: hide(b_:=1) extrapoline (0,abs(tx_)) of e_[i_]
     else: extrapolate (0,abs(tx_)) of e_[i_] fi;
    ei__:=if is_line(e_[i__]):
     (1/(1-abs(ty_)))[point 1 of e_[i__], point 0 of e_[i__] ] -
     point 1 of e_[i__]
    elseif ty_<0: hide(b__:=1) extrapoline (abs(ty_),1) of e_[i__]
    else: extrapolate (abs(ty_),1) of e_[i__] fi;
% clumsy HEURESIS (choosing an optimal intersection point, if there are
% more intersections):
    save t_; (t_.a1,length(ei__)-t_.b1)=ei_ intersectiontimes reverse ei__;
    if t_.a1>0:
     ei_1:=if (known b_) and (t_.a1>1):
       force_convex_edge(point 0 of e_[i_], postcontrol 0 of e_[i_],
        precontrol 1 of e_[i_], point t_.a1 of ei_)
      else: subpath (0,t_.a1) of ei_ fi;
     ei__1:=if (known b__) and (t_.b1<1):
      force_convex_edge(point t_.b1 of ei__, postcontrol 0 of e_[i__],
       precontrol 1 of e_[i__], point 1 of e_[i__])
      else: subpath (t_.b1,infinity) of ei__ fi;
     (length(ei_)-t_.a2,t_.b2)=reverse ei_ intersectiontimes ei__;
     if length((t_.a1,t_.b1)-(t_.a2,t_.b2))>eps:
      ei_2:=if (known b_) and (t_.a2>1):
       force_convex_edge(point 0 of e_[i_], postcontrol 0 of e_[i_],
        precontrol 1 of e_[i_], point t_.a2 of ei_)
       else: subpath (0,t_.a2) of ei_ fi;
      ei__2:=if (known b__) and (t_.b2<1):
      force_convex_edge(point t_.b2 of ei__, postcontrol 0 of e_[i__],
       precontrol 1 of e_[i__], point 1 of e_[i__])
       else: subpath (t_.b2,infinity) of ei__ fi;
      if arclength(ei_1)+arclength(ei__1) > arclength(ei_2)+arclength(ei__2):
       ei_1:=ei_2; ei__1:=ei__2;
      fi
     fi
     e_[i_]:=ei_1; e_[i__]:=ei__1;
    fi
   fi
  fi
 endfor
 for i_:=0 upto l_-1:
  hide(i__:=(i_-1) mod l_)
  if cycle p or (i_>0):
   if length((point 1 of e_[i__])-(point 0 of e_[i_]))>1/4ignore_nib_limit:
   % the constant |1/4ignore_nib_limit| plays a similar role
   % to that of the |SNAP_TO_NODE| variable in pf2mt1.awk
    (point 1 of e_[i__])
    if known local_tip_@#(i_): – else:
     && pen_join(predir 1 of e_[i__],postdir 0 of e_[i_],point i_ of p,
      local_nib_@#(i_)) &&
    fi
   fi
  fi
  % reconstruct |e_[i_]| (possibly ignoring direction(s)):
  (point 0 of e_[i_])
   if is_line(e_[i_]):
   % the using of |–| circumvents \MF{}/\MP{} instable behaviour:
   % the operator |...| may cause that a control point and a node
   % (nearly) coincide (note that this is feature, not a bug);
   % thus, it is advisable for |pen_stroke_method=0|; supposedly,
   % it is also adequate for |pen_stroke_method=1|:
    –
   else:
    if pen_stroke_method=0:
     if not ignore_dir_(i_): {postdir 0 of e_[i_]} fi ...
     if not ignore_dir_(i_+1): {predir 1 of e_[i_]} fi
    elseif (pen_stroke_method=1) or (pen_stroke_method=2):
     .. controls (postcontrol 0 of e_[i_]) and (precontrol 1 of e_[i_]) ..
    fi
   fi
 endfor
 if cycle p: cycle else: (point 1 of e_[l_-1]) fi
enddef;
newinternal pen_stroke_method;

% Macro |pen_stroke| performs an operation known as "expanding stroke";
% we'll call the result of the operation a "pen envelope" (for
% a given path). The macro has one optional parameter, |opts| (|text|),
% and two obligatory ones: input path |p| (|expr|)
% and a |result| (|suffix|). A user has an access to subpaths of the
% envelope, namely: |result.r| is the right edge of the envelope,
% |result.l|—its left edge, |result.b|—is a fragment of the pen outline
% joining left and right edge of the envelope at the beginning
% node of the path, |result.e|—is a similar fragment at the ending
% node of the path (see the picture below). If the path |p|
% is cyclic, then |result.e| and |result.b| are undefined,
% otherwise the variable |result| contains additionally the complete
% expanded stroke.

% For finding an envelope, a default path (|default_nib|, returned
% by |fix_nib|) is used except nodes for which the parameter |opts|
% sets another pen. Mastering the usage of the parameter |opts| allows
% a user to achieve nontrivial effects. The parameter |opts| is a list
% (space-separated or semicolon-separated) of the following
% operators: (1) |nib|, (2) |cut|, (3) |tip|, and (4) |ignore_directions|.

% Ad 1. The macro |nib| has two parameters:
% |nib|(pen)(list_of_nodes), where "pen" is a path returned by
% macro |fix_nib|, and "list_of_nodes" contains comma-separated numbers
% (times) of the nodes of the path |p| at which a given pen is to be
% used. If needed, the outline is complemented at corner nodes
% with a fragment of a pen path. Such a join corresponds to the setting
% |linejoin:=rounded| in \MP{}. If the path |p| is non-cyclic,
% its ends are also complemented with appropriate fragments of a pen path
% (the setting |linecap:=rounded|). Such a method of joining is also applied
% by |pen_stroke| to nodes not mentioned in the parameter |opts|.
% The result of the following statement
% \LINE{\descriptioncomments
% |pen_stroke(nib(default_nib xyscaled (1,2))(infinity))(p)(q)|
% \unskip}
% \descriptioncomments
% that changes the pen at the last node of the path,
% is shown in the following picture:
% \LINE{\epsfbox{\illusname.110}}

% Ad 2. The call of the macro |cut| has the form: |cut|(angle,
% pen)(list_of_nodes) or |cut|(pen, angle)(list_of_nodes),
% where "pen" and "list_of_nodes" are defined as
% previously. The pen parameter can be omitted which means using a default
% pen (|default_nib|). The macro replaces a default pen with a special
% “razor” pen at specified nodes. More precisely, it is a projection of a
% given pen in the direction of the path |p| at a given node onto a
% straight line going through this node under the angle specified in the
% respective parameter of the macro. Uf\/f\/f\dots\ The angle of the straight
% line can be defined either absolutly (with respect to the axis |x|)
% or—by adding a prefix ‘|rel|’—relatively to the direction of the path
% at a given node. From the point of view of a user, the result of the
% macro |cut| is “cutting” the expanded stroke with a straight
% line. This operation is particularly useful at the ends of a path and
% corresponds to the setting |linecap:=butt| in \MP{}, except that in \MP{}
% one cannot specify angles. The result of the statement
% \LINE{\descriptioncomments
% |pen_stroke(cut(45)(0)|
% |cut(default_nib xyscaled (1,2), rel 90)(infinity))(p)(q)|
% \unskip}
% \descriptioncomments
% that cuts both ends and, moreover, changes a pen
% at the ending node is shown in the figure below
% (at the beginning node, the absolute angle of 45 degrees is specified,
% at the ending one—the relative angle of 90 degrees):

% Ad 3. The call of the macro |tip| has the form |tip|(pen,
% pre_elongate, post_elongate)(list_of_nodes), where “pen”
% and “list_of_nodes” have the same meaning as previously.
% In particular, a pen can be omitted. At corner nodes
% specified in the list of nodes, the consecutive elements of the outline
% are not joined with an appropriate subpath of a pen; instead, they
% are elongated (extrapolated) until they intersect. This process corresponds
% (roughly) to the \MP{} setting |linejoin:=mitered|:

% The illustration above is the result of the following call
% of the macro |pen_stroke| (the macro |tip| is invoked with default
% settings, only the number of a node is specified):
% |pen_stroke(tip()(3))(p)(q); draw q;|
% The optional parameters |pre_elongation| and |post_elongation| define how
% far the consecutive edges (segments) should be elongated in order to make
% them intersect each other (the measure is the time). If one parameter is
% omitted, both will receive the same value; if both are omitted, a default
% value, |(0.5,0.5)| (it corresponds to elongation by circa 50\%), will be
% used. The precise meaning of the pre- and post-elongation is defined as
% follows: for a given pre-edge |e1|, post-edge |e2|, pre-elongation |v1|
% and post-elongation |v2|, the paths
% |extrapolate (0, 1/(1+v1) of e1| and
% |extrapolate (v2/(1+v2), 1) of e2| are computed
% (i.e., for the default elongation: |extrapolate (0, 2/3) of s1|
% and |extrapolate (1/3, 1) of s2|, respectively).
% If elongated curves do not intersect, the terminal nodes
% of the consecutive segments are joined with a straight line. The latter
% property can be used to obtain a result corresponding to the \MP{} setting
% |linejoin:=beveled|: it suffices to apply a null elongation, i.e.,
% |tip|(0)(list_of_nodes). Changing the first (empty) parameter
% of the |tip| macro in the previous example would yield the following
% result:

% Ad 4. The macro |ignore_directions| has a different character. It is
% invoked with one parameter being a comma-separated list of nodes:
% |ignore_directions|(list_of_nodes). The numbers {\it must be\/} followed
% by sufixes |l| or |r|. The macro causes that, at specified nodes,
% the direction of the outline is not forced to be parallel to the direction
% of the path |p| (which is the default); instead, the direction is
% calculated by \MP{}. Suffixes determine whether the direction
% is not to be forced at the right (|r|) or the left (|l|) edge (with
% respect to the direction of the path |p|). This heuristic
% trick can be used to improve the appearence of the outline
% if the “inner” part of the envelope has too tight arcs.
%% The examples of the usage of this macro can be found in the \MP{} version
%% of D. E. Knuth’s ‘logo’ font (letters ‘P’ and ,S’).

vardef pen_stroke(text opts)(expr p)(suffix result) =
 forsuffixes $=,r,l,b,e:
  if not path result$: scantokens("path " & genericize(str result$)); fi
 endfor
 save a_, a___, d_, i_, k_, n_, p_, z_, norm_, norml_, normr_, normlr_,
  fix_opts_, ignore_dir_, ignore_dir___, local_nib_, local_nib___,
  local_tip_, default_tip_, local_tip___, % internal
  all, rel, last, nib, cut, tip, ignore_directions, current_node; % exported
 numeric ignore_dir___[\\]; pair default_tip_, local_tip___[\\];
 path local_nib___[\\];
 pair a_, d_, z_[\\]; path p_;
%% xpart norm_ norml_ normr_ normlr_
 vardef norm_ primary n =
  if cycle p: n mod last else: if n<0: 0 elseif n>last: last else: n fi fi
 enddef;
 vardef norml_ primary n = -norm_ n -1 enddef;
 vardef normr_ primary n = norm_ n +1 enddef;
 vardef normlr_@# primary i= if str @#="l": -norm_(last-i)-1 else: i+1 fi enddef;
 last=length(p);
 vardef rel primary a =
  angle((gendir current_node of p) slant_stroke)+a
 enddef;
 def all =
  hide(% locally we use the prefix rather than postfix noitation;
       % a trick due to the |suffix| parameter of the |allcont_| macro
   vardef l primary n = (norml_ n,0) enddef;
   vardef r primary n = (normr_ n,0) enddef) allcont_
 enddef;
 def allcont_ suffix $ =
  $0 for i_:=1 upto last if cycle p: -1 fi: , $i_ endfor
 enddef;
 vardef fixopts_(suffix optname)(text nodes) text val =
%% intersectiontimes lcont_ rcont_
  save l, r, lcont_, rcont_;
  def l = lcont_ whatever enddef; primarydef a lcont_ b = (norml_ a,0) enddef;
  def r = rcont_ whatever enddef; primarydef a rcont_ b = (normr_ a,0) enddef;
  for n_:=nodes:
   if numeric n_:
    current_node:=norm_ n_;
    optname[norml_ n_]:=optname[normr_ n_]
   else:
    current_node:=abs(xpart n_)-1; % the inverse of both |norml_| and |normr_|
    optname[xpart(n_)]
   fi :=val; % |val| may depend on |current_node|
  endfor
 enddef;
 def nib(text nib_)(text nodes) = % nib and node list
  fixopts_(local_nib___)(nodes)
   begingroup
    p_:=default_nib; for k_:=nib_: p_:=k_; endfor \\ p_
   endgroup;
 enddef;
 def cut(text nib_and_ang)(text nodes) = % angle, nib and node list
  fixopts_(local_nib___)(nodes)
   begingroup
    p_:=default_nib;
    for k_:=nib_and_ang:
     if numeric k_: a_:=dir(unslant_angle(k_)); else: p_:=k_; fi
    endfor
    d_:=gendir current_node of p;
    z_1:=whatever*a_=tangent_point(d_,p_)+whatever*d_;
    z_2:=whatever*a_=tangent_point(-d_,p_)+whatever*d_;
    z_1--z_2
   endgroup;
 enddef;
 def tip(text nib_and_lim)(text nodes)= % limit(s) and node list
  i_:=0; for n_:=nib_and_lim: if numeric n_: i_[incr i_]:=n_; fi endfor
  fixopts_(local_tip___)(nodes)
   elongation_to_times(if i_=0: default_elongation, default_elongation
    elseif i_=1: i_1, i_1 else: i_1, i_2 fi);
  fixopts_(local_nib___)(nodes)
   begingroup
    p_:=default_nib; for k_:=nib_and_lim: if path k_: p_:=k_; fi endfor \\ p_
   endgroup;
  enddef;
  def ignore_directions(text nodes) = % node list
   fixopts_(ignore_dir__)(nodes) 1;
  enddef;
  if default_cap=0:
   if not cycle p: cut(rel 90)(0,last); fi
  elseif default_cap=1: % do nothing
  else:
   errhelp ”Admissible values are 0, 1; continue, I’ll use the value 1.”;
   errmessage ”PX: improper ‘default_cap’ value (”&decimal(default_cap)&”)”;
  fi
  opts;

  if default_join=0:
   default_tip_:=elongation_to_times(default_elongation, default_elongation);
  elseif default_join=1: % no tip setting, do nothing
  elseif default_join=2:
   default_tip_:=(1,0); % |(1,0)=elongation_to_times(0,0)|
  else:
   errhelp ”Admissible values are 0, 1, 2; continue, I’ll use the value 1.”;
   errmessage ”PX: improper ‘default_join’ value (”&decimal(default_join)&”)”;
  fi
  vardef ignore_dir_@#(expr i) = known ignore_dir__[normlr_@# i] enddef;
  vardef local_tip_@#(expr i) = if known local_tip__[normlr_@# i]:
   local_tip__[normlr_@# i] else: default_tip_ fi enddef;
  vardef local_nib_@#(expr i) = if known local_nib__[normlr_@# i]:
   local_nib__[normlr_@# i] else: default_nib fi enddef;
  result.r:=pen_stroke_edge.r(p);
  result.l:=pen_stroke_edge.l(reverse p);
  if not cycle p:
   result.b:=pen_cap(predir infinity of result.l,postdir 0 of result.r,
    -postdir 0 of p,point 0 of p,local_nib_.l(last),local_nib_.r(0));
   result.e:=pen_cap(predir infinity of result.r,postdir 0 of result.l,
    predir last of p,point last of p,local_nib_.r(last), local_nib_.l(0));
   result:=result.r && result.e && result.l && result.b && cycle;
  fi
 enddef;

 vardef pen_cap(expr a,b,c,p,niba,nibb)=
  if path_eq(niba,nibb): pen_join(a,b,p,niba)
  else: pen_join(a,c rotated 90,p,niba)--pen_join(c rotated 90,b,p,nibb)
  fi
 enddef;

```

Postscript Font Generation

```
% POSTSCRIPT FONT GENERATION

% Note that this has been stripped down a lot from the METATYPE1
% original code; most of the stuff for hinting, ligature tables,
% METAFONT-style proof generation, and so on has been removed
% because it’s irrelevant to Tsukurimashou.

vardef pfi_file = jobname & ”.pfi” enddef;
vardef pic_file = ”piclist” enddef;
vardef dim_file = jobname & ”.dim” enddef;

errorstopmode; warningcheck:=-1;
ignore:=whatever; process:=0; utilize:=1; store:=2; % constants for introducing
let semicolon_=; ; % stores original meaning of a semicolon
newinternal tracingdimens; % if |tracingdimens>0| then |dim_file| is generated

def write_special = % additional info to be processed by AWK
 special ”%GLYNFO: ” &
enddef;
vardef mtone_glyph_pfx = ”MT1: glyph ” & str glyph_name & ”: ” enddef;
def mtone_message = message mtone_glyph_pfx & enddef;

% Macro write_tex provides contact with the
% outer world. The macro contains the information about EPSes that is
% used for proofing and assembling the font; must be consistent with
% the definitions contained in the files ‘mpform.sty’ and ‘mp2pf.awk’.
vardef write_tex(expr name, num) =
 write ”\EPSNAMEandNUMBER{” & name & ”}{” & decimal(num) & ”}”
  to pic_file & ”.tex”
enddef;

% The following macros are related to the operation of slanting.
% In particular, they enable to keep a fixed width of a stem
% after slanting.
vardef slant_ang = % should be rather called “local_slant_angle”
 slang \\ if known glyph_slanting.glyph_name: * glyph_slanting.glyph_name fi
enddef;
vardef slant_val = tand(slant_ang) enddef;
vardef slant_preadjust(expr slope, slang) =
% |if sind(angle(slope))=0: 1 else:|
% | abs(sind(angle(slope))/sind(angle(cotd(angle(slope))+tand(slang),1)))|
% |fi|
% Correction of stem size taking into account its slope and a slant angle;
% nice formula, isn’t it? Much simpler than the previous one, yet equivalent:
 length(unitvector(slope) slanted tand(slang))
enddef;
```

```
vardef slant_stroke_val = slant_val enddef; % compatibility with plain_ex.mp

vardef stem_corr (expr slope) = slant_preadjust(slope, slant_ang) enddef;

def italicized = % fairly complex operation
 if slang<>0:
  if known glyph_slanting.glyph_name:
   if glyph_slanting.glyph_name=0: shifted (math_axis*tand(slang),0) fi
  fi
  shifted (italic_shift*tand(slang),0) % re-positioning
  slanted slant_val % and slanting
 fi
enddef;

primarydef b || c =
 whatever*b + c*stem_corr(b)*unitvector(b rotated 90)
enddef;

primarydef c /\ b =
% A variant of the |leg| procedure that iteratively counteracts slant
% deformation; as with |leg|, given: |c| — hypotenuse (vector) of
% a right-angled triangle, |b| — the length of one of its legs;
% result: the other leg of the triangle (vector),
 if slant_ang=0: (c leg b)
 else:
  begingroup save b_, b___, n_; b_:=b___:=b; n_:=10;
   forever:
    b_:=b*stem_corr(c leg b_);
    exitif (abs(b_-b___)<.01) or (n_<=0);
    b___:=b_; n_:=n_-1;
   endfor
   if (abs(b_-b___)>=.01):
    errhelp ”The result is likely to be weird.”;
    errmessage mtone_glyph_pfx & ”iteration hasn’t converged”;
   fi
   c leg b_
  endgroup
 fi
enddef;

% Obsolete?
vardef rib(expr t,p,r) text u = % |u| is either empty or a vector
 save k_; pair k_; for i_:=u: k_:=u; endfor
 if unknown k_: k_=((udir t of p) rotated 90); fi
 (point t of p) + r * k_ * stem_corr(k_ rotated 90)
enddef;

% The operation {\it compose_path\/} is useful in \MP{} programs
% automatically generated from PFB sources (pf2mt1 utility). Suffixes
% $a$ and $b$ of control nodes stand for ‘after’ and ‘before’, respectively;
% The operation {\it compose_path\/} makes use of the operation
% {\it compose_segment\/} that serves for constructing non-cyclic
% paths. Undefined nodes are ignored.
vardef compose_segment@#(expr m,n) = % |m<=n|, not checked
 if unknown inside_compose_path_: save idx_, n_; n_:=-1; fi
 save n___; n___=n_+1;
 for i_:=m upto n: if known @#[i_]: idx_[incr(n_)]=i_; fi endfor
 for i_:=n___ upto n_-1:
   @#[idx_[i_]] .. controls
         @#[idx_[i_]] if known @#[idx_[i_]]a: a fi
    and @#[idx_[i_+1]] if known @#[idx_[i_+1]]b: b fi
    ..
 endfor
 @#[idx_[n_]]
enddef;
vardef compose_path@#(expr n) =
 save inside_compose_path_, idx_, n_; n_:=-1; inside_compose_path_:=1;
 compose_segment@#(0,n)
  if @#[idx_[0]]=@#[idx_[n_]]: & else: -- fi \\ cycle
enddef;

% Basic macros for building character glyphs:
vardef round_node_values(expr p) =
 save d_; % candidates for Flex -- no checking for "straightlinessness"
 for t_=0 upto length(p)-1:
  if round(point t_ of p)=round(point t_+1 of p):
   hide(mtone_message "degenerated bezier " & ", length=" &
    decimal(length(p)) & " " & ", time=" & decimal(t_) & " ";
   show p)
  else:
   round(point t_ of p)..
    if if known d_[t_] or known d_[t_+1]: false else:
     is_line(subpath (t_,t_+1) of p) fi:
    controls round(point t_ of p) and round(point t_+1 of p)
   else:
    controls round(postcontrol t_ of p) and round(precontrol t_+1 of p)
   fi
   ..
  fi
 endfor
 round(point length(p) of p) \\ if cycle p: & cycle fi
enddef;

primarydef a start b =
 if cycle a:
  if b=default: default_start_(a)
  else: ((subpath (b,length(a)+b) of a) & cycle) fi
 else: a fi
enddef;

newinternal default; default:=infinity;
vardef default__start_(expr p) =
 save i__,j__,pi__,pj__; pair pi__,pj__;
 j__:=0; pj__:=point j__ of p;
 for i__=1 upto length(p):
  pi__:=point i__ of p;
  if (xpart(pi__)>xpart(pj__)) or
   (xpart(pi__)=xpart(pj__)) and (ypart(pi__)<ypart(pj__)):
   j__:=i__; pj__:=point j__ of p;
  fi
 endfor
 (subpath (j__, length(p)+j__) of p) & cycle
enddef;

def Fill text glist =
 begingroup
  save h__; path h__;
  for g__:=glist:
   h__:=g__ start.default; % JMN's suggestion
   if turningnumber h__<>1:
    errhelp ”The result is likely to be weird.”;
    errmessage mtone_glyph__pfx & ”strange turning number in Fill, ” &
     decimal(turningnumber h__);
   fi
   if glyph__usage div store = 1: % storing
    glyph__stored.glyph__name[incr glyph__stored.glyph__name.num]=h__;
   fi
   glyph__list[incr glyph__list.num]:=round__node__values(h__ italicized);
   update__glyph__bb(glyph__list[glyph__list.num]);
  endfor;
 endgroup
enddef;

def unFill text glist =
 begingroup
  save h__; path h__;
  for g__:=glist:
   h__:=g__ start.default; % JMN's suggestion
   if turningnumber h__<>-1:
    errhelp ”The result is likely to be weird.”;
    errmessage mtone_glyph__pfx & ”strange turning number in unFill, ” &
     decimal(turningnumber h__);
   fi
   if glyph__usage div store = 1: % storing
```

```
    glyph_stored.glyph_name[incr glyph_stored.glyph_name.num]=h_;
   fi
   glyph_list[incr glyph_list.num]:=round_node_values(h_ italicized);
  endfor;
 endgroup
enddef;

def fix_hsbw (expr xr,ml,mr) =
 glyph_shift:=round(ml); % shift = left margin
 glyph_width:=round(xr+ml+mr); % declared width plus margins
 if glyph_usage div store = 1: % storing
  glyph_shift.glyph_name:=glyph_shift; glyph_width.glyph_name:=glyph_width;
 fi
enddef;

def fix_exact_hsbw(expr xr,ml,mr) =
 glyph_shift:=round(ml); % shift = left margin
 glyph_width:=xr+ml+mr; % declared width plus margins
 if glyph_usage div store = 1: % storing
  glyph_shift.glyph_name:=glyph_shift; glyph_width.glyph_name:=glyph_width;
 fi
enddef;

% Macros below set PostScript and \TeX{} units; a trick with `\#'
% in {\it tfm\_units\/} proves useful in achieving compatibility
% with the Knuthian fonts (e.g., it is employed in {\it logo\/} font).
% Old versions of {\it tfm\_units\/} and {\it ps\_units\/} are less
% accurate, but are kept because of backward compatibility reasons.
vardef tfm_units(text x) =
 save #; if known (x#): x# else: x/(1000/designsize) fi
enddef;
vardef old_tfm_units(text x) =
 save #; if known (x#): x# else: x/1000*designsize fi
enddef;

vardef ps_units(expr x) = x*(1000/designsize) enddef;
vardef old_ps_units(expr x) = x/designsize*1000 enddef;

def define_ps_units(text t) =
 forsuffixes $:=t: $:=ps_units($.#); endfor
enddef;
def define_whole_ps_units(text t) =
 forsuffixes $:=t: $:=round(ps_units($.#)); endfor
enddef;
def define_even_ps_units(text t) =
 forsuffixes $:=t: $:=2round(1/2ps_units($.#)); endfor
enddef;

```

```
% In general, all objects are supposed to be drawn by the
% {\bf endglyph} macro, i.e., all drawing operations are deferred.
% The same concerns labelling, which necessitates redefinition
% of labelling macros.

def label_(suffix pos)(expr s,z, dot_or_not) =
% should be more complex if overlapping labels are to be avoided
enddef;
string label_defaultfont; label_defaultfont:=”cmr10”;
newinternal label_defaultscale; label_defaultscale:=magstep 5;

% If the {\it project\/} variable is assigned value greater than 0,
% proofing mode is assumed; the following macros display then
% the details of the construction of glyphs for proofing purposes.
% The larger value of the variable {\it project}, the more details
% are visualised.
def local_drawoptions (text t) = % to be used within a group, see below
% \begingroup \def\\#1{{\it#1}}% local: no underscore hacks
 save _op_; drawoptions(t);
% \endgroup
enddef;

def update_glyph_bb(expr p) =
 if unknown glyph_llx:
  glyph_llx:=xpart(llcorner(p)); glyph_lly:=ypart(llcorner(p));
  glyph_urx:=xpart(urcorner(p)); glyph_ury:=ypart(urcorner(p));
 else:
  if xpart(llcorner(p))<glyph_llx: glyph_llx:=xpart(llcorner(p)); fi
  if ypart(llcorner(p))<glyph_lly: glyph_lly:=ypart(llcorner(p)); fi
  if xpart(urcorner(p))>glyph_urx: glyph_urx:=xpart(urcorner(p)); fi
  if ypart(urcorner(p))>glyph_ury: glyph_ury:=ypart(urcorner(p)); fi
 fi
enddef;
string stencil_dir;
def ship_glyphs =
 begingroup
  local_drawoptions();
  for g_:=1 upto glyph_list.num:
   if turningnumber glyph_list[g_]>0: fill else: unfill fi
    glyph_list[g_] shifted (glyph_shift,0);
  endfor
 endgroup
enddef;
newinternal show_stroke_size; show_stroke_size:=1.5;
color show_stroke_color; show_stroke_color:=red;

color label_dot_color, label_text_color;
label_dot_color:=.8white; label_text_color:=black;
```

```
newinternal label_dot_size; label_dot_size:=3bp;

% Begin and end of the definitions of a character glyph:
def begin_skip =
 let endglyph = fi;
 let ; = end_skip semicolon_
 if false:
enddef;
def end_skip =
 let ; = semicolon_ semicolon_
 let endglyph = endglyph_;
enddef;

def uni_name(text name) = % name is either a suffix or a string expression
  if is_suffix(name):
    name
  else:
    scantokens (begingroup
      save rval;
      string rval;
      rval:=”” for i=1 upto length (name):
        & ”_” & (substring (i-1,i) of (name))
      endfor;
      rval
    endgroup)
  fi
enddef;

def glyph_name_ext = enddef;
def beginglyph(text name) =
 %
 def original_glyph_name = name enddef;
 def glyph_name = uni_name(name) glyph_name_ext enddef; % to use in |endglyph|
 numeric glyph_usage; glyph_usage:=glyph_usage.glyph_name;
 if unknown glyph_usage: expandafter begin_skip fi
 string ps_name; ps_name:=original_glyph_name;
 if unknown ps_name:
  errhelp ”Use macro ‘introduce’ or ‘assign_name’ prior to ‘beginglyph.’”;
  errmessage ”MT1: PS name not assigned to ” & str glyph_name;
 fi
 if name_used(original_glyph_name):
  errhelp ”Proceed if you wish, I’ll use the second glyph description.”;
  errmessage ”MT1: double output: name ” & (str glyph_name);
 fi
 if glyph_usage mod store = 1: % utilizing
  mark_name_used(original_glyph_name);
 fi
 numeric glyph_code, glyph_num; glyph_code:=name_to_code(original_glyph_name);
```

```
  if glyph_code<0: glyph_num:=500-decr(min_glyph_code); else:
   glyph_num:=100+glyph_code;
   if glyph_code>max_glyph_code: max_glyph_code:=glyph_code; fi
  fi
 %
 beginfig(glyph_num)
 if glyph_usage mod store = 1: % utilizing
  write_special "NAME " & ps_name & " " & decimal(glyph_code);
   % mpform.sty and mp2pf.awk interface
% |write_tex(glyph_name, glyph_num);|
   write_tex(ps_name, glyph_num);
 fi;
 glyph_list.num:=label_list.num:=0;
 path glyph_list[\\];
 picture label_list[\\]; pair label_list.dot[\\];
 numeric glyph_llx, glyph_lly, glyph_urx, glyph_ury;
 numeric bitmap_scale; pair bitmap_offset;
 numeric glyph_shift, glyph_width, glyph_axis;
 save glyph;
 hstem_list.num:=vstem_list.num:=hstem_list.cov:=vstem_list.cov:=0;
 pair hstem_list[\\], vstem_list[\\];
 path hstem_list_segms[\\], vstem_list_segms[\\];
 numeric old_hinting_scheme, new_hinting_scheme;
 if glyph_usage div store = 1: % storing
  if not path glyph_stored.glyph_name[0]: % glyph_name may contain digits
   scantokens("path " & genericize(str glyph_stored.glyph_name) & "[]");
   scantokens("pair " & genericize(str hstem_stored.glyph_name) & "[]");
   scantokens("path " & genericize(str hstem_stored_segms.glyph_name) & "[]");
   scantokens("pair " & genericize(str vstem_stored.glyph_name) & "[]");
   scantokens("path " & genericize(str vstem_stored_segms.glyph_name) & "[]");
  fi
  glyph_stored.glyph_name.num:=0;
  hstem_stored.glyph_name.num:=0; vstem_stored.glyph_name.num:=0;
 fi
 scantokens extra_beginglyph;
enddef;

picture endglyph_picture;
def endglyph =
 scantokens extra_endglyph;
 % usually, |currentpicture=nullpicture|, but if not (i.e., some
 % extra objects have been drawn), the picture must be shifted:
 endglyph_picture:=currentpicture shifted (glyph_shift,0);
 currentpicture:=nullpicture;
 if known glyph_axis: % actually, used only with stored chars
  glyph_axis.glyph_name:=glyph_axis;
 fi
% fix char dimensions and write them to TFM and/or |dim_file|
% independently of |glyph_usage| (|dim_file|)
% fix_tfm_data(glyph_urx+glyph_shift, glyph_ury);
 if glyph_usage mod store = 1: % utilizing
    write_special "HSBW * " & decimal(glyph_width);
    write_special "BEGINCHAR";
    ship_glyphs;
  endfig;
 else:
  endgroup; % ends figure without shipping it out
 fi
enddef;
let endglyph_=endglyph;
string extra_beginglyph, extra_endglyph; extra_beginglyph=extra_endglyph="";

% Additional macros
vardef fix_name_list text t =
 string name_list[]; numeric name_list.num; name_list.num:=0;
 save , ; let , = fix_name_list_; fix_name_list_ t
enddef;
def fix_name_list_ suffix name =
 ; % important semicolon!
 if str name<>"": fix_name_list_s_ name else: fix_name_list_e_ "_" & fi
enddef;
def fix_name_list_s_ suffix s_name = fix_name_list_e_ (str s_name) enddef;
def fix_name_list_e_ expr e_name = % name is expected to be of the string type
 name_list[incr name_list.num]=e_name
enddef;

def introduce suffix name =
 if str name="": introduce_
 elseif (substring (0,1) of str name)<>"_": introduce_ name
 else: introduce__ name fi
enddef;
def introduce_ expr name = % name is expected to be a string expression
 introduce__ uni_name(name)
enddef;
vardef introduce__@#(expr usage, slanting)(text stencil) =
 if (unknown process_selected or known process_selected@#)
  and known usage and unknown ignore_selected@#:
  glyph_usage@#:=usage; % |ignore=whatever|, |process=0|, |utilize=1|, |store=2|
  if unknown glyph_ps_name@#: % set default:
   assign_name @# (substring (1,infinity) of (str @#));
  fi
  glyph_slanting@#:=slanting; % ignore |slant_ang| if |0|; use |slant_ang| otherwise
  % |stencil| can be either string (recommended) or suffix (with default
  % extension |".eps"| — obsolete), hence some trickery below
  save r_; string r_;
  for i_:=stencil: if string i_: r_:=i_; fi endfor
  if unknown r_:
   forsuffixes i_:=stencil: r_:= str i_; endfor
   if r_<>””: r_:=r_ & ”.eps”; fi
  fi
  if r_<>””:
   if not string glyph_stencil@#:
    scantokens(”string ” & genericize(str glyph_stencil@#));
   fi
   glyph_stencil@# = r_;
  fi
 fi
enddef;

vardef assign_name@#(expr ps_name) =
 if not string glyph_ps_name @#:
  scantokens(”string ” & genericize(str glyph_ps_name@#));
 fi
 glyph_ps_name@#:=ps_name;
enddef;

def standard_introduce(expr name) =
 introduce name (utilize+store)(1)();
enddef;

vardef name_to_code(text name) =
 save res_, name_; string name_;
 name_:=name; res_=-1;
 for i:=0 upto 255: % 1-to-1 coding presumed
  if known code_to_name_[i]: if code_to_name_[i]=name_: res_:=i; fi fi
  exitif res_>-1;
 endfor
 res_
enddef;
def encode(text name)(expr glyph_code)=
 if (glyph_code<0) or (glyph_code>255):
  errhelp ”The code must be within the range 0..255.”;
  errmessage ”MT1: improper code ” & decimal(glyph_code) &
   ” (‘encode’ ignored)”;
 elseif known code_to_name_[glyph_code]:
  errhelp ”A given code can be assigned only to one name (obviously).”;
  errmessage ”MT1: repeated code for ” & code_to_name_[glyph_code] &
    ” (” & decimal(glyph_code) & ”; ‘encode’ ignored)”;
 else:
  code_to_name_[glyph_code]:=name;
 fi
enddef;
string code_to_name_[\\];

```

```
vardef name_used(text name) =
 save res_, name_; boolean res_; string name_;
 name_:=name; res_:=false;
 for i:=1 upto max_name_used: res_:=(name_used_[i]=name_); exitif res_; endfor
 res_
enddef;
def mark_name_used(text name)=
 name_used_[incr max_name_used]:=name;
enddef;
string name_used_[\\]; newinternal max_name_used;

vardef string_date =
 if day<10: "0" & fi decimal(day) & "." &
 if month<10: "0" & fi decimal(month) & "." &
 decimal(year)
enddef;

def set_pfi (suffix kind) (expr val) =
 if known val:
  if (numeric val) or (string val) or (boolean val):
   if (numeric val) and (not numeric pf_info_set.kind):
    scantokens ("numeric " & genericize(str pf_info_set.kind));
   elseif (string val) and (not string pf_info_set.kind):
    scantokens ("string " & genericize(str pf_info_set.kind));
   elseif (boolean val) and (not boolean pf_info_set.kind):
    scantokens ("boolean " & genericize(str pf_info_set.kind));
   fi
   pf_info_set.kind:=val;
    write str kind & " : " &
     if string val: val
     elseif numeric val: decimal(val)
     elseif boolean val: if val: "true" else: "false" fi
     fi
     to pfi_file;
  else:
   errhelp "Proceed, I'll just ignore the setting.";
   errmessage "MT1: pf_info keys can only be of numeric, string " &
    "and boolean type";
  fi
 fi
enddef;

def pf_info_version expr v = set_pfi(VERSION,v); enddef;

def pf_info_creationdate text t =
 begingroup
  save k_; k_:=0;
  for t_:=t: k_:=k_+1; set_pfi(CREATION_DATE, t_); exitif k_=1; endfor
  if k_=0: set_pfi(CREATION_DATE, string_date); fi
 endgroup
enddef;

def pf_info_fontname text t =
 begingroup
  save k_; k_:=0;
  for t_:=t: k_:=k_+1;
   if k_=1: set_pfi(FONT_NAME, t_); fi
   if k_=2: set_pfi(FULL_NAME, t_); fi
   exitif k_=2;
  endfor
  if k_=1: set_pfi(FULL_NAME, pf_info_set.FONT_NAME); fi
 endgroup
enddef;

def pf_info_author expr v = set_pfi(AUTHOR,v); enddef;
% There is ‘much ado about nothing,’ i.e., about the sign of descender:
% in a PFB file in an ‘ADL’ comment, descender is positive, while in an AFM
% in a ‘Descender’ comment — negative; we will distinguish between
% the two, the more so as ‘ADL’ comment is not mentioned in
% in the Adobe documentation {\it Adobe Type 1 Font Format}.

def pf_info_ascender expr v = ascender:=v; set_pfi(ASCENDER,v); enddef;
def pf_info_descender expr v = descender:=v; set_pfi(DESCENDER,v); enddef;

def pf_info_adl text t =
 begingroup
  save k_; k_:=0;
  for t_:=t: k_:=k_+1;
   if (k_=1) and known t_: adl_ascender:=t_; set_pfi(ADL_ASCENDER,t_); fi
   if (k_=2) and known t_: adl_descender:=t_; set_pfi(ADL_DESCENDER,t_); fi
   if (k_=3) and known t_: adl_lineskip:=t_; set_pfi(ADL_LINESKIP,t_); fi
   exitif k_=3;
  endfor
 endgroup
enddef;

def pf_info_underline text t =
 begingroup
  save k_; k_:=0;
  for t_:=t: k_:=k_+1;
   if k_=1: set_pfi(UNDERLINE_POSITION,t_); fi
   if k_=2: set_pfi(UNDERLINE_THICKNESS,t_); fi
   exitif k_=2;
  endfor
 endgroup
enddef;
```

```

def pf_info_pfm text t =
% parameters: name, bold (0 or 1), italic (0 or 1), char set;
% each of them can be either known or unknown or ”*” (which means unknown);
% the last parameter can be either numeric or string representation of
% a valid Perl numeric value (e.g., ”0xFF” means 255).
 begingroup
  save k_; k_:=0;
  for t_:=t: k_:=k_+1;
   if (k_=1) and known t_: set_pfi(PFM_NAME,t_); fi
   if (k_=2) and known t_: set_pfi(PFM_BOLD,t_); fi
   if (k_=3) and known t_: set_pfi(PFM_ITALIC,t_); fi
   if (k_=4) and known t_: set_pfi(PFM_CHARSET,t_); fi
   exitif k_=4;
  endfor
 endgroup
enddef;

def pf_info_fixedpitch expr v = set_pfi(FIXED_PITCH,v); enddef;
def pf_info_capheight expr v = uc_height:=v; set_pfi(CAPHEIGHT,v); enddef;
def pf_info_weight expr v = set_pfi(WEIGHT,v); enddef;
def pf_info_stdvstem expr v = set_pfi(STDVW,v); enddef;
def pf_info_stdhstem expr v = set_pfi(STDHW,v); enddef;
def pf_info_forcebold expr v = set_pfi(FORCE_BOLD,v); enddef;

% TeX-related font info (fontdimens and headerbytes):
def pf_info_fontdimen text t = % exceptionally, TFM units expected
 begingroup
  save i_, k_, b_; boolean b_;
  k_:=0;
  if true for t_:=t: hide(k_:=k_+1) and known t_ endfor and (k_<=3):
   k_:=0;
   for t_:=t: k_:=k_+1;
    if k_=1:
     i_:=t_;
     % |b| means “we are ready to override (possibly) the previous value
     % of a font parameter unless we are inside |complete_tfm_info| and
     % then we want to set only a ‘virgin’ value.”
     b_:=unknown completing_tfm_info or unknown pf_info_set.FONT_DIMEN[i_];
    fi
    if b_ and (k_=2): set_pfi(FONT_DIMEN[i_],t_); fontdimen i_: t_; fi
    if b_ and (k_=3): set_pfi(DIMEN_NAME[i_],t_); fi
   endfor
   if b_ and (k_=2): set_pfi(DIMEN_NAME[i_],”(unknown fontdimen name)”); fi
  else:
   errhelp ”Proceed, I’ll just ignore TFM fontdimen settings.”;
   errmessage ”MT1: invalid TFM fontdimen data”;
  fi
 endgroup
enddef;
def pf_info_headerbyte text t =
 begingroup
  save i_, k_; k_:=0;
  if true for t_:=t: hide(k_:=k_+1) and known t_ endfor and (k_=2):
   k_:=0;
   for t_:=t: k_:=k_+1;
    if k_=1: i_:=t_; fi
    if k_=2:
     set_pfi(HEADER_BYTE[i_],if numeric t_: decimal(t_) else: t_ fi);
     if i_=9: % encoding scheme, e.g., |"TEX TEXT"|
      headerbyte 9: BCPL_string(t_,40); fi
     if i_=49: % font family, e.g., |"CMR"|
      headerbyte 49: BCPL_string(t_,20); fi
     if i_=72: % family member number, which should be between 0 and 255
      headerbyte 72: t_; fi
    fi
   endfor
  else:
   errhelp "Proceed, I'll just ignore TFM headerbyte settings.";
   errmessage "MT1: invalid TFM headerbyte data";
  fi
 endgroup
enddef;
def pf_info_designsize expr v = % |designsize| is special
 designsize:=v; set_pfi(DESIGN_SIZE,decimal(v) & " (in points)");
enddef;
def pf_info_italicangle expr v =
 begingroup
  save tfm_units; vardef tfm_units(text x) = c enddef;
  slang:=v; set_pfi(ITALIC_ANGLE,-v);
  pf_info_fontdimen 1, if known slant: slant else: tand(slang) fi, "(slant)";
 endgroup
enddef;
def pf_info_space text t = % three in one
 begingroup
  save k_; k_:=0;
  for t_:=t: k_:=k_+1;
   if (designsize<>0) and known t_:
    if k_=1:
     space:=t_; pf_info_fontdimen 2, tfm_units(space), "(space)";
    elseif k_=2:
     space_stretch:=t_; pf_info_fontdimen 3, tfm_units(space_stretch),
      "(space stretch)";
    elseif k_=3:
     space_shrink:=t_; pf_info_fontdimen 4, tfm_units(space_shrink),
       "(space shrink)";
    fi
   fi
   exitif k_=3;
  endfor
 endgroup
enddef;
def pf_info_normal_space text t =
 begingroup
  save k_; k_:=0;
  if true for t_:=t: hide(k_:=k_+1) endfor and (k_<=2):
   k_:=0;
   for t_:=t: k_:=k_+1;
    if (k_=1) and known t_: space:=t_; fi
    if (k_=2) and known t_: % |t_| is expected to be in TFM units
     pf_info_fontdimen 2, t_, ”(space)”;
    fi
   endfor
   if (k_=1) and (designsize<>0) and known space:
    pf_info_fontdimen 2, tfm_units(space), ”(space)”;
   fi
  fi
 endgroup
enddef;
def pf_info_space_stretch text t =
 begingroup
  save k_; k_:=0;
  if true for t_:=t: hide(k_:=k_+1) endfor and (k_<=2):
   k_:=0;
   for t_:=t: k_:=k_+1;
    if (k_=1) and known t_: space_stretch:=t_; fi
    if (k_=2) and known t_: % |t_| is expected to be in TFM units
     pf_info_fontdimen 3, t_, ”(space stretch)”;
    fi
   endfor
   if (k_=1) and (designsize<>0) and known space_stretch:
    pf_info_fontdimen 3, tfm_units(space_stretch), ”(space stretch)”;
   fi
  fi
 endgroup
enddef;
def pf_info_space_shrink text t =
 begingroup
  save k_; k_:=0;
  if true for t_:=t: hide(k_:=k_+1) endfor and (k_<=2):
   k_:=0;
   for t_:=t: k_:=k_+1;
    if (k_=1) and known t_: space_shrink:=t_; fi
    if (k_=2) and known t_: % |t_| is expected to be in TFM units
```

```
     pf_info_fontdimen 4, t_, "(space shrink)";
    fi
   endfor
   if (k_=1) and (designsize<>0) and known space_shrink:
    pf_info_fontdimen 4, tfm_units(space_shrink), "(space shrink)";
   fi
  fi
 endgroup
enddef;
def pf_info_xheight text t =
 begingroup
  save k_; k_:=0;
  if true for t_:=t: hide(k_:=k_+1) endfor and (k_<=2):
   k_:=0;
   for t_:=t: k_:=k_+1;
    if (k_=1) and known t_: lc_height:=t_; set_pfi(XHEIGHT, t_); fi
    if (k_=2) and known t_: % |t_| is expected to be in TFM units
     pf_info_fontdimen 5, t_, "(xheight)";
    fi
   endfor
   if (k_=1) and (designsize<>0) and known lc_height:
    pf_info_fontdimen 5, tfm_units(lc_height), "(xheight)";
   fi
  fi
 endgroup
enddef;
def pf_info_quad text t =
 begingroup
  save k_; k_:=0;
  if true for t_:=t: hide(k_:=k_+1) endfor and (k_<=2):
   k_:=0;
   for t_:=t: k_:=k_+1;
    if (k_=1) and known t_: quad:=t_; fi
    if (k_=2) and known t_: % |t_| is expected to be in TFM units
     pf_info_fontdimen 6, t_, "(quad)";
    fi
   endfor
   if (k_=1) and (designsize<>0) and known quad:
    pf_info_fontdimen 6, tfm_units(quad), "(quad)";
   fi
  fi
 endgroup
enddef;
def pf_info_extra_space text t =
 begingroup
  save k_; k_:=0;
  if true for t_:=t: hide(k_:=k_+1) endfor and (k_<=2):
   k_:=0;
   for t_:=t: k_:=k_+1;
    if (k_=1) and known t_: extra_space:=t_; fi
    if (k_=2) and known t_: % |t_| is expected to be in TFM units
     pf_info_fontdimen 7, t_, ”(extra space)”;
    fi
   endfor
   if (k_=1) and (designsize<>0) and known extra_space:
    pf_info_fontdimen 7, tfm_units(extra_space), ”(extra space)”;
   fi
  fi
 endgroup
enddef;
def pf_info_encoding text t =
 begingroup
  save k_; k_:=0;
  for t_:=t: k_:=k_+1;
   if (k_=1) and known t_: if t_<>””: set_pfi(ENCODING_SCHEME, t_); fi fi
   if (k_=2) and known t_: if t_<>””: pf_info_headerbyte 9, t_; fi fi
   if (k_=3) and known t_: if t_<>””: set_pfi(ENCODING_NAME, t_); fi fi
   exitif k_=3;
  endfor
  if (k_=1) and known pf_info_set.ENCODING_SCHEME % upward compatibility
   and unknown pf_info_set.HEADER_BYTE9:
   pf_info_headerbyte 9, pf_info_set.ENCODING_SCHEME;
  fi
 endgroup
enddef;
def pf_info_familyname text t =
 begingroup
  save k_; k_:=0;
  for t_:=t: k_:=k_+1;
   if k_=1: set_pfi(FAMILY_NAME, t_); fi
   if k_=2: pf_info_headerbyte 49, t_; fi
   exitif k_=2;
  endfor
  if k_=1: pf_info_headerbyte 49, pf_info_set.FAMILY_NAME; fi
 endgroup
enddef;

% bluezz forever...
newinternal blue_fuzz, blue_scale, blue_shift;
blue_fuzz:=0; % Adobe Type 1 Font Format, p. 41
blue_scale:=0.0454545;
blue_shift:=7;

% it is advisable to avoid typso whenever possible:
def show_compose expr x = show_compose_ :=x; enddef;
def show_fills expr x = show_fills_ :=x; enddef;
def show_strokes expr x = show_strokes_ :=x; enddef;
def show_paths expr x = show_paths_ :=x; enddef;
def show_labels expr x = show_labels_ :=x; enddef;
def show_boxes expr x = show_boxes_ :=x; enddef;
def show_stems expr x = show_stems_ :=x; enddef;
def show_stencils expr x = show_stencils_ :=x; enddef;

string extra_beginfont, extra_endfont; extra_beginfont=extra_endfont="";

def beginfont =
 min_glyph_code=max_glyph_code=0;
 complete_param_setting;
 scantokens extra_beginfont;
enddef;

def complete_param_setting =
 if designsize=0: designsize:=10; fi
 if unknown space: space:=333; fi
 if unknown space_stretch: space_stretch:=round(1/2space); fi
 if unknown space_shrink: space_shrink:=round(1/3space); fi
 if unknown extra_space: extra_space:=round(1/3space); fi
 if unknown quad: quad:=1000; fi
 if unknown slang:
  if known slant: % compatibility with the Old Tradition...
   slang:=angle(1, slant);
  else: slang:=0; fi
 fi
 if unknown uc_height: uc_height:=750; fi
 if unknown lc_height: lc_height:=400; fi
 if unknown italic_shift: italic_shift:=-40; fi % used to be |-100|
 if unknown depth: depth:=-250; fi
 if unknown ascender: ascender:=uc_height; fi
 if unknown descender: descender:=depth; fi
 if unknown adl_ascender: adl_ascender:=uc_height; fi
 if unknown adl_descender: adl_descender:=-depth; fi
 if unknown adl_lineskip: adl_lineskip:=0; fi
 if unknown top_line: top_line:=adl_ascender+1/2adl_lineskip; fi
 if unknown bot_line: bot_line:=-(adl_descender+1/2adl_lineskip); fi
 if unknown math_axis: math_axis:=250; fi
 if unknown math_rule: math_rule:=40; fi
 begingroup
  save rth_, pt_, subs_, desc_depth_, fig_height_, asc_height_;
  rth_:=math_rule; pt_:=100;
  % math symbol font parameters (defaults excerpted from cmsy10)
  subs_:=7/10;
  desc_depth_:=70/36pt_; fig_height_:=232/36pt_; asc_height_:=250/36pt_;
  if unknown num_one:
   num_one:=math_axis+3.51rth_+54/36pt_+subs_*desc_depth_; fi
  if unknown num_two: num_two:=math_axis+1.51rth_+30/36pt_; fi
  if unknown num_three: num_three:=math_axis+1.51rth_+48/36pt_; fi
  if unknown denom_one:
   denom_one:=3.51rth_+subs_*fig_height_+124/36pt_-math_axis; fi
  if unknown denom_two:
   denom_two:=1.51rth_+subs_*fig_height_+30/36pt_-math_axis; fi
  if unknown sup_one: sup_one:=8.99pt_-subs_*asc_height_; fi
  if unknown sup_two: sup_two:=8.49pt_-subs_*asc_height_; fi
  if unknown sup_three: sup_three:=104/36pt_; fi
  if unknown sub_one: sub_one:=54/36pt_; fi
  if unknown sub_two: sub_two:=-8.49pt_+2subs_*asc_height_+3.1rth_; fi
  if unknown sup_drop: sup_drop:=subs_*asc_height_-36/36pt_; fi
  if unknown sub_drop: sub_drop:=18/36pt_; fi
  if unknown delim_one: delim_one:=23.9pt_; fi
  if unknown delim_two: delim_two:=10.1pt_; fi
  % math extension font parameters (defaults excerpted from cmex10)
  if unknown big_op_spacing_one: big_op_spacing_one:=40/36pt_; fi;
  if unknown big_op_spacing_two: big_op_spacing_two:=60/36pt_; fi;
  if unknown big_op_spacing_three: big_op_spacing_three:=72/36pt_; fi;
  if unknown big_op_spacing_four: big_op_spacing_four:=216/36pt_; fi;
  if unknown big_op_spacing_five: big_op_spacing_five:=36/36pt_; fi;
 endgroup;
enddef;

def endfont =
 scantokens extra_endfont;
 complete_pf_info;
 complete_tfm_info;
 scantokens ”end”;
enddef;

def complete_pf_info =
 if unknown pf_info_set.DESIGN_SIZE: pf_info_designsize designsize; fi
 if unknown pf_info_set.VERSION: pf_info_version ”0.000”; fi
 if unknown pf_info_set.AUTHOR: pf_info_author ”Unknown”; fi
 if unknown pf_info_set.CREATION_DATE: pf_info_creationdate; fi
 if unknown pf_info_set.FAMILY_NAME: pf_info_familyname ”Untitled”; fi
 if unknown pf_info_set.FONT_NAME: pf_info_fontname ”Untitled”; fi
 if unknown pf_info_set.ASCENDER: pf_info_ascender ascender; fi
 if unknown pf_info_set.DESCENDER: pf_info_descender descender; fi
 if unknown pf_info_set.ADL_ASCENDER:
  pf_info_adl adl_ascender, whatever, whatever;
 fi
 if unknown pf_info_set.ADL_DESCENDER:
  pf_info_adl whatever, adl_descender, whatever;
 fi
 if unknown pf_info_set.ADL_LINESKIP:
  pf_info_adl whatever, whatever, adl_lineskip;
 fi
 if unknown pf_info_set.UNDERLINE_POSITION: pf_info_underline -200, whatever; fi
 if unknown pf_info_set.UNDERLINE_THICKNESS: pf_info_underline whatever, math_rule; fi
 if unknown pf_info_set.ITALIC_ANGLE: pf_info_italicangle slang; fi
 if unknown pf_info_set.FIXED_PITCH: pf_info_fixedpitch false; fi
 if unknown pf_info_set.CAPHEIGHT: pf_info_capheight uc_height; fi
 if unknown pf_info_set.XHEIGHT: pf_info_xheight lc_height; fi
 if unknown pf_info_set.WEIGHT: pf_info_weight "Normal"; fi
 if unknown pf_info_set.STDVW: fi % just ignore
 if unknown pf_info_set.STDHW: fi % just ignore
 if unknown pf_info_set.FORCE_BOLD: pf_info_forcebold false; fi
 if unknown pf_info_set.ENCODING_SCHEME:
  pf_info_encoding "FontSpecific", whatever;
 fi
 if unknown pf_info_set.HEADER_BYTE9:
  pf_info_encoding whatever, "UNSPECIFIED";
 fi
 if unknown pf_info_set.BLUE_VALUES: set_pfi(BLUE_VALUES, ""); fi
 if unknown pf_info_set.OTHER_BLUES: fi % just ignore
 if unknown pf_info_set.BLUE_FUZZ: set_pfi(BLUE_FUZZ, blue_fuzz); fi
 if unknown pf_info_set.BLUE_SCALE: set_pfi(BLUE_SCALE, blue_scale); fi
 if unknown pf_info_set.BLUE_SHIFT: set_pfi(BLUE_SHIFT, blue_shift); fi
 % for those who like smart (implicit) systems:
 if unknown no_implicit_spaces:
  if not name_used("space"):
   if unknown glyph_usage._space: introduce _space (utilize)(0)(); fi;
   if (name_to_code("space")<0) and (unknown code_to_name_32):
    encode("space") (32);
   fi
   beginglyph("_space") fix_hsbw(space,0,0); endglyph;
  fi
  if not name_used("nbspace"):
   if unknown glyph_usage._nbspace: introduce _nbspace (utilize)(0)(); fi;
   %
   beginglyph("_nbspace") fix_hsbw(space,0,0); endglyph; % normal space width
  fi
 fi
enddef;

def complete_tfm_info =
 % complete fontdimen info:
 % |designsize| is expected to be known
 % |slant| dimen has already been set; |xheight| dimen -- not necessarily,
 % but |pf_info_set.XHEIGHT| is known:
 completing_tfm_info:=1;
 pf_info_xheight whatever,
   if known lc_height#: lc_height# else: tfm_units(pf_info_set.XHEIGHT) fi;
 pf_info_normal_space space if known space#: , space# fi;
 pf_info_space_stretch space_stretch
  if known space_stretch#: , space_stretch# fi;
 pf_info_space_shrink space_shrink if known space_shrink#: , space_shrink# fi;
 pf_info_quad quad if known quad#: , quad# fi;
 pf_info_extra_space extra_space if known extra_space#: , extra_space# fi;
  font_math_rule math_rule;
  font_math_axis math_axis;
 % complete header info:
 pf_info_headerbyte 72, max(0, 254 - round 2designsize);
 completing_tfm_info:=whatever;
enddef;

def BCPL_string(expr s,n)= % string |s| becomes an |n|-byte BCPL string
 for l:=if length(s)>=n: n-1 else: length(s) fi: l
  for k:=1 upto l: , substring (k-1,k) of s endfor
  for k:=l+2 upto n: , 0 endfor endfor
enddef;

% The Old Tradition...
def font_size expr x = designsize:=x enddef;
def font_slant expr x = fontdimen 1: x enddef;
def font_normal_space expr x = fontdimen 2: x enddef;
def font_normal_stretch expr x = fontdimen 3: x enddef;
def font_normal_shrink expr x = fontdimen 4: x enddef;
def font_x_height expr x = fontdimen 5: x enddef;
def font_quad expr x = fontdimen 6: x enddef;
def font_extra_space expr x = fontdimen 7: x enddef;

% A New Tradition...
def def_font_param (suffix param_name)(expr param_num, param_desc) =
 def param_name text x =
  begingroup save #; % cf. the definition of |tfm_units|
   if (known x#) or ((designsize<>0) and known x):
    pf_info_fontdimen param_num, tfm_units(x), "(" & param_desc & ")";
   fi
  endgroup
 enddef;
enddef;

def_font_param (font_math_rule, 8, "math rule");
def_font_param (font_math_axis, 22, "math axis");
% symbol fonts
def_font_param (font_num_one, 8, "num1");
def_font_param (font_num_two, 9, "num2");
def_font_param (font_num_three, 10, "num3");
def_font_param (font_denom_one, 11, "denom1");
def_font_param (font_denom_two, 12, ”denom2”);
def_font_param (font_sup_one, 13, ”sup1”);
def_font_param (font_sup_two, 14, ”sup2”);
def_font_param (font_sup_three, 15, ”sup3”);
def_font_param (font_sub_one, 16, ”sub1”);
def_font_param (font_sub_two, 17, ”sub2”);
def_font_param (font_sup_drop, 18, ”sup_drop”);
def_font_param (font_sub_drop, 19, ”sub_drop”);
def_font_param (font_delim_one, 20, ”delim1”);
def_font_param (font_delim_two, 21, ”delim2”);
% extension fonts
def_font_param (font_big_op_spacing_one, 9, ”big_op_spacing1”);
def_font_param (font_big_op_spacing_two, 10, ”big_op_spacing2”);
def_font_param (font_big_op_spacing_three, 11, ”big_op_spacing3”);
def_font_param (font_big_op_spacing_four, 12, ”big_op_spacing4”);
def_font_param (font_big_op_spacing_five, 13, ”big_op_spacing5”);

endinput
```

obstack.mp

```
%
% Object stack for Tsukurimashou
% Copyright (C) 2011, 2012 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(obstack);

```

Object Stack Data

```
% OBJECT STACK DATA

% object types:
% ”anchor” - uses transform, numeric
% ”hook” - uses string, numeric
% ”lcblob” - uses path p
% ”null” - uses nothing
% ”pbox” - uses transform
% ”stroke” - uses path p, path q, numeric array, bool array

vardef init_obstack =
  numeric obstacktype[];
  numeric obstackn[];
  numeric obstackna[][];
  numeric obstacknaa[][][];
  boolean obstackb[];
  boolean obstackba[][];
  path obstackp[];
  path obstackq[];
  transform obstackt[];
  string obstacks[];
  numeric sp;
  sp:=1;
enddef;

sp:=0;

% numeric values, needed for syntax reasons

def boalternate = 1924 enddef;
def bokeepshape = 1838 enddef;
def boserif = 1746 enddef;
def bosize = 1393 enddef;
def botip = 1322 enddef;
def botoexpand = 1972 enddef;

def hsmain_render = 1304 enddef;

def otanchor = 1882 enddef;
def othook = 1753 enddef;
def otlcblob = 1722 enddef;
def otnull = 1699 enddef;
def otpbox = 1007 enddef;
def otstroke = 1069 enddef;
```

Object Stack Methods

```
% OBJECT STACK METHODS

vardef expand_pbox =
  begingroup
    save mysp,i;
    numeric mysp;
    for i=sp-1 downto 1:
      if (obstacktype[i]=otpbox) and (known obstackba.botoexpand[i]):
        if obstackba.botoexpand[i]:
          mysp:=i;
        fi;
      fi;
      exitif known mysp;
    endfor;
    if known mysp:
      obstackba.botoexpand[mysp]:=false;
% message ”expanding ” & decimal mysp;
      save x,y,myxf;
      numeric x[],y[];
      transform myxf;
      z1=(70,830);
      z2=(930,-30);
      for i=mysp+1 upto sp-1:
        if obstacktype[i]=otpbox:
          x3:=xpart ((0,0.5) transformed obstackt[i]);
          if x3<x1: x1:=x3; fi;
          y3:=ypart ((0.5,1) transformed obstackt[i]);
          if y3>y1: y1:=y3; fi;
          x4:=xpart ((1,0.5) transformed obstackt[i]);
          if x4>x2: x2:=x4; fi;
          y4:=ypart ((0.5,0) transformed obstackt[i]);
          if y4<y2: y2:=y4; fi;
        fi;
      endfor;
      z0=(x1,y2);
      (0,0) transformed myxf=z0+(-20,-20);
      (0,1) transformed myxf=z1+(-20,20);
      (1,0) transformed myxf=z2+(20,-20);
      obstackt[mysp]:=myxf;
    else:
      errmessage ”Can’t find PBOX to expand”;
    fi;
  endgroup;
  perl_structure:=perl_structure&”],”;
enddef;

vardef find_stroke(expr idx) =
  (find_whatever(otstroke,idx))
enddef;

vardef find_whatever(expr w,idx) =
  begingroup
    save i,j;
    numeric i,j;
    i:=sp-1;
    j:=-idx;
    forever:
      exitif i<=0;
      if obstacktype[i]=w:
        j:=j-1;
      fi;
      exitif j<0;
      i:=i-1;
    endfor;
    i
  endgroup
enddef;

vardef get_bosize(expr idx) =
  obstackna.bosize[find_whatever(otstroke,idx)]
enddef;

vardef get_anchor_with_default(expr atype,default_anchor) =
  begingroup
    save i,j;
    numeric i,j;
    i:=0;
    forever:
      j:=find_whatever(otanchor,i);
      exitif j<=0;
      exitif obstackn[j]=atype;
      i:=i-1;
    endfor;
    if j<=0: default_anchor else: obstackt[j] fi
  endgroup
enddef;

vardef get_anchor(expr atype) =
  get_anchor_with_default(atype,identity)
enddef;

vardef get_lcblob(expr idx) =
  obstackp[find_whatever(otlcblob,idx)]
enddef;

vardef get_strokep(expr idx) =
  obstackp[find_whatever(otstroke,idx)]
enddef;

vardef get_strokeq(expr idx) =
  obstackq[find_whatever(otstroke,idx)]
enddef;

vardef pop_hook =
  obstacktype[find_whatever(othook,0)]:=otnull;
enddef;

vardef pop_lcblob =
  obstacktype[find_whatever(otlcblob,0)]:=otnull;
enddef;

vardef pop_stroke =
  obstacktype[find_whatever(otstroke,0)]:=otnull;
enddef;

vardef push_anchor(expr atype,anchor) =
  obstacktype[sp]:=otanchor;
  obstackn[sp]:=atype;
  if pair anchor:
    obstackt[sp]:=identity shifted anchor;
  else:
    obstackt[sp]:=anchor;
  fi;
  sp:=sp+1;
enddef;

vardef push_hook(expr stage,htext) =
  obstacktype[sp]:=othook;
  obstackn[sp]:=stage;
  obstacks[sp]:=htext;
  sp:=sp+1;
enddef;

vardef push_lcblob(expr blob) =
  obstacktype[sp]:=otlcblob;
  obstackp[sp]:=blob;
  sp:=sp+1;
enddef;

vardef push_pbox(expr pbname) =
  obstacktype[sp]:=otpbox;
  obstackt[sp]:=identity scaled 900 shifted (50,-50);
  obstacks[sp]:=pbname;
  sp:=sp+1;
enddef;

vardef push_pbox_explicit(expr pbname,pbox) =
  obstacktype[sp]:=otpbox;
  obstackt[sp]:=pbox;
  obstacks[sp]:=pbname;
  sp:=sp+1;
enddef;

vardef push_pbox_toexpand(expr pbname) =
  obstacktype[sp]:=otpbox;
  obstackt[sp]:=identity scaled 1000 shifted (0,-100);
  obstacks[sp]:=pbname;
  obstackba.botoexpand[sp]:=true;
% message ”to expand ” & decimal sp;
  sp:=sp+1;
  perl_structure:=perl_structure&”[’”&pbname&”’,”;
enddef;

vardef push_stroke(expr p,q) =
  obstacktype[sp]:=otstroke;
  obstackp[sp]:=p;
  obstackq[sp]:=q;
  obstackna.bosize[sp]:=100;
  sp:=sp+1;
  perl_structure:=perl_structure&”push_stroke,”;
enddef;

vardef replace_lcblob(expr idx)(text blob) =
  begingroup
    save oldblob;
    path oldblob;
    oldblob:=obstackp[find_whatever(otlcblob,idx)];
    obstackp[find_whatever(otlcblob,idx)]:=blob;
  endgroup;
enddef;

vardef replace_strokep(expr idx)(text curves) =
  begingroup
    save oldp;
    path oldp;
    oldp:=obstackp[find_whatever(otstroke,idx)];
    obstackp[find_whatever(otstroke,idx)]:=curves;
  endgroup;
  perl_structure:=perl_structure&”replace_strokep,”;
enddef;

vardef replace_strokeq(expr idx)(text curves) =
  begingroup
    save oldq;
    path oldq;
    oldq:=obstackq[find_whatever(otstroke,idx)];
    obstackq[find_whatever(otstroke,idx)]:=curves;
  endgroup;
enddef;

vardef set_boalternate(expr idx) =
  obstackba.boalternate[find_stroke(idx)]:=true;
enddef;

vardef set_bokeepshape(expr idx) =
  obstackba.bokeepshape[find_whatever(otlcblob,idx)]:=true;
enddef;

vardef set_boserif(expr idx,t,srf) =
  obstacknaa.boserif[find_stroke(idx)][t]:=srf;
enddef;

vardef set_bosize(expr idx,bos) =
  obstackna.bosize[find_stroke(idx)]:=bos;
  if bos=0:
    perl_structure:=perl_structure&”bosize0,”;
  fi;
enddef;

vardef set_botip(expr idx,t,bt) =
  obstacknaa.botip[find_stroke(idx)][t]:=bt;
enddef;
```

frac-intro.mp

```
%
% Common code for Tsukurimashou fractions
% Copyright (C) 2011 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(fracintro);

```

```

transform nxf[];

frac.in.x1=200;
frac.in.x2=800;
frac.in.y1=latin_wide_baseline;
frac.in.y2=latin_wide_top;

frac.one.y1=0.02[frac.in.y1,frac.in.y2];
frac.one.y2=0.40[frac.in.y1,frac.in.y2];
frac.one.y3=0.51[frac.in.y1,frac.in.y2];
frac.one.y4=0.60[frac.in.y1,frac.in.y2];
frac.one.y5=0.98[frac.in.y1,frac.in.y2];

(frac.one.x1+frac.one.x2)/2=500;
frac.one.x2-frac.one.x1=320;

frac.two.y1=0.04[frac.in.y1,frac.in.y2];
frac.two.y2=0.38[frac.in.y1,frac.in.y2];
frac.two.y3=0.51[frac.in.y1,frac.in.y2];
frac.two.y4=0.62[frac.in.y1,frac.in.y2];
frac.two.y5=0.96[frac.in.y1,frac.in.y2];

(frac.two.x1+frac.two.x3)/2=500;
(frac.two.x3-frac.two.x2)=
  (frac.two.x2-frac.two.x1);
frac.two.x3-frac.two.x1=600;

frac.three.y1=0.06[frac.in.y1,frac.in.y2];
frac.three.y2=0.36[frac.in.y1,frac.in.y2];
frac.three.y3=0.51[frac.in.y1,frac.in.y2];
frac.three.y4=0.64[frac.in.y1,frac.in.y2];
frac.three.y5=0.94[frac.in.y1,frac.in.y2];

(frac.three.x1+frac.three.x4)/2=500;
(frac.three.x4-frac.three.x3)=
  (frac.three.x3-frac.three.x2)=
  (frac.three.x2-frac.three.x1);
frac.three.x4-frac.three.x1=700;

frac.four.y1=0.08[frac.in.y1,frac.in.y2];
frac.four.y2=0.34[frac.in.y1,frac.in.y2];
frac.four.y3=0.51[frac.in.y1,frac.in.y2];
frac.four.y4=0.66[frac.in.y1,frac.in.y2];
frac.four.y5=0.92[frac.in.y1,frac.in.y2];

(frac.four.x1+frac.four.x5)/2=500;
(frac.four.x5-frac.four.x4)=
  (frac.four.x4-frac.four.x3)=
  (frac.four.x3-frac.four.x2)=
  (frac.four.x2-frac.four.x1);
frac.four.x5-frac.four.x1=800;

frac.half.y1=0.10*latin_vcentre;
frac.half.y2=0.82*latin_vcentre;
frac.half.y3=latin_vcentre;
frac.half.y4=1.18*latin_vcentre;
frac.half.y5=1.90*latin_vcentre;

(frac.half.x1+frac.half.x2)/2=250;
frac.half.x2-frac.half.x1=330;

vardef hexdig(expr d) =
  if d<10: decimal d else: char (d+87) fi
enddef;

vardef make_digit_set(expr xfm,thispage,place) =
  numeric ccount;
  ccount:=0;
  forsuffixes i=zero,one,two,three,four,five,six,seven,eight,nine:
    begintsuglyph(”uFF” & thispage & place & hexdig(ccount),
        hex(place & hexdig(ccount)));
      tsu_xform(xfm)(numeral.i);
      tsu_render;
    endtsuglyph;
    ccount:=ccount+1;
  endfor;
enddef;
```

latin-intro.mp

```
%
% Shared code for Tsukurimashou latin
% Copyright (C) 2011, 2012 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(latinintro);

```

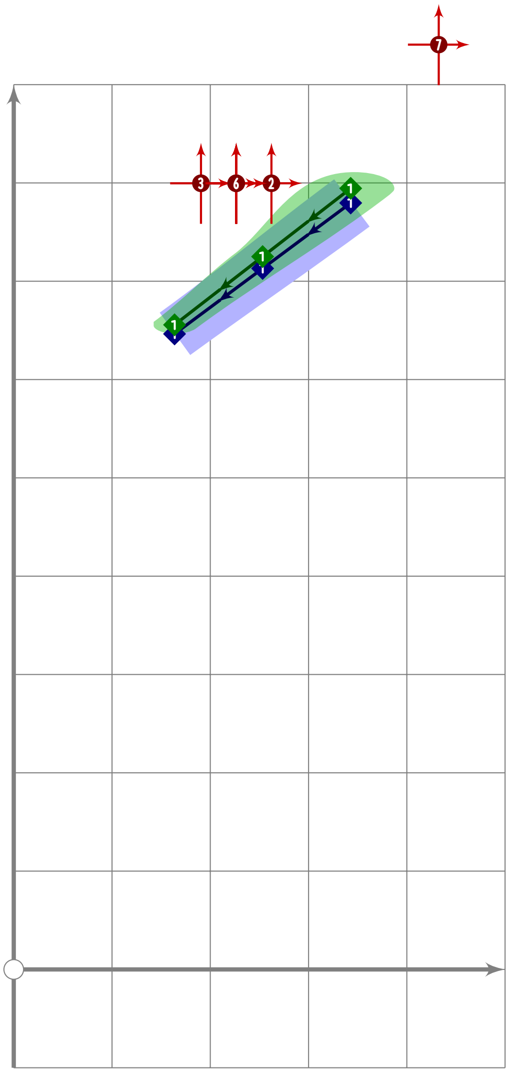
```

latin_wide_low_h:=latin_wide_baseline+mbrush_height*0.8;
latin_wide_high_h:=latin_wide_top-mbrush_height*0.8;
latin_wide_low_r:=latin_wide_baseline+mbrush_height*0.8-7;
latin_wide_high_r:=latin_wide_top-mbrush_height*0.8+15;
if sharp_corners:
  latin_wide_low_v:=latin_wide_baseline;
  latin_wide_high_v:=latin_wide_top;
else:
  latin_wide_low_v:=latin_wide_baseline+mbrush_height*0.8;
  latin_wide_high_v:=latin_wide_top-mbrush_height*0.8;
fi;

vardef vmetric(expr a) =
  (a[latin_wide_low_h,latin_wide_high_h])
enddef;

latin_wide_xheight:=vmetric(0.65);
latin_wide_xheight_h:=latin_wide_xheight-mbrush_height*0.6;
latin_wide_xheight_r:=latin_wide_xheight-mbrush_height*0.6+15;
if sharp_corners:
  latin_wide_xheight_v:=latin_wide_xheight;
else:
  latin_wide_xheight_v:=latin_wide_xheight-mbrush_height*0.5;
fi;

latin_wide_desc:=vmetric(-0.35);
latin_wide_desc_h:=latin_wide_desc+mbrush_height*0.6;
latin_wide_desc_r:=latin_wide_desc+mbrush_height*0.6-10;
if sharp_corners:
  latin_wide_desc_v:=latin_wide_desc;
else:
  latin_wide_desc_v:=latin_wide_desc+mbrush_height*0.5;
fi;

latin_wide_lc_baselift:=vmetric(0.02);

```

```
transform tsu_xf.accentedcap,tsu_xf.cap_upper_accent,tsu_xf.low_centre_accent;

if is_proportional:
  tsu_xf.accentedcap=identity;
else:
  xpart tsu_xf.accentedcap=1;
  xypart tsu_xf.accentedcap=yxpart tsu_xf.accentedcap=0;
  (500,vmetric(0)) transformed tsu_xf.accentedcap=(500,vmetric(0));
  (500,vmetric(1)) transformed tsu_xf.accentedcap=(500,vmetric(0.82));
fi;

accent_default[anc_upper]=identity shifted (500,vmetric(0.75));
accent_default[anc_grave]=identity
  shifted (500+0.4*tsu_punct_size,vmetric(0.75));
accent_default[anc_acute]=identity
  shifted (500-0.4*tsu_punct_size,vmetric(0.75));
accent_default[anc_wide]=identity xscaled 0.75 shifted (500,vmetric(0.75));
accent_default[anc_tilde]=identity xscaled 0.75 shifted (500,vmetric(0.75));
accent_default[anc_ring]=identity shifted (500,vmetric(0.75));
accent_default[anc_caron_comma]=identity shifted (730,vmetric(0.93));
accent_default[anc_lower]=identity shifted (500,vmetric(-0.26));
accent_default[anc_lower_connect]=identity shifted (500,vmetric(0));
accent_default[anc_centre]=identity
  scaled ((latin_wide_high_r-latin_wide_low_r)/200) shifted centre_pt;

% this one is for capitals that have NOT been shrunk
tsu_xf.cap_upper_accent=identity shifted (500,vmetric(1.10));

tsu_xf.low_centre_accent=identity
    scaled ((latin_wide_xheight_r-latin_wide_low_r)/200)
    shifted (xpart centre_pt,(latin_wide_xheight_r+latin_wide_low_r)/2);

vardef tsu_accent.shift_anchors(text c)(expr s) =
  begingroup;
    save killflag;
    boolean killflag;
    for i:=1 upto max_accent_seen:
      if unknown accent_has_default[i]:
        killflag:=true;
      elseif not accent_has_default[i]:
        killflag:=true;
      else:
        killflag:=false;
        for j=sp-1 downto 1:
          if obstacktype[j]=otanchor:
            if obstackn[j]=i:
              killflag:=true;
            fi;
          fi;
          exitif killflag;
        endfor;
      fi;
      if not killflag:
        def ai = i enddef;
        def olda = ((0,0) transformed accent_default[i]) enddef;
        if c:
          push_anchor(i,accent_default[i]);
        fi;
      fi;
    endfor;
  endgroup;
  for i:=sp-1 downto 1:
    if obstacktype[i]=otanchor:
      begingroup
        def ai = obstackn[i] enddef;
        def olda = ((0,0) transformed obstackt[i]) enddef;
        if c:
          obstackt[i]:=obstackt[i] shifted s;
        fi;
      endgroup;
    fi;
  endfor;
enddef;
```

```
vardef tsu_accent.up_default_anchors =
  tsu_default_anchor(anc_upper,accent_default[anc_upper]
    shifted (0,vmetric(1.10)-vmetric(0.75)));
  tsu_default_anchor(anc_grave,accent_default[anc_grave]
    shifted (0,vmetric(1.10)-vmetric(0.75)));
  tsu_default_anchor(anc_acute,accent_default[anc_acute]
    shifted (0,vmetric(1.10)-vmetric(0.75)));
  tsu_default_anchor(anc_wide,identity xscaled 1.2
    transformed accent_default[anc_wide]
    shifted (0,vmetric(1.10)-vmetric(0.75)));
  tsu_default_anchor(anc_tilde,accent_default[anc_tilde]
    shifted (0,vmetric(1.10)-vmetric(0.75)));
  tsu_default_anchor(anc_ring,accent_default[anc_ring]
    shifted (0,vmetric(1.10)-vmetric(0.75)));
  tsu_default_anchor(anc_caron_comma,
    accent_default[anc_caron_comma] shifted (200,0));
  tsu_default_anchor(anc_lower,accent_default[anc_lower]);
  tsu_default_anchor(anc_lower_connect,accent_default[anc_lower_connect]);
  tsu__default__anchor(anc__centre,accent__default[anc__centre]);
enddef;

vardef  tsu__accent.low_default_anchors =
  tsu__default__anchor(anc__upper,accent__default[anc__upper]);
  tsu__default__anchor(anc__grave,accent__default[anc__grave]);
  tsu__default__anchor(anc__acute,accent__default[anc__acute]);
  tsu__default__anchor(anc__wide,accent__default[anc__wide]);
  tsu__default__anchor(anc__tilde,accent__default[anc__tilde]);
  tsu__default__anchor(anc__ring,accent__default[anc__ring]);
  tsu__default__anchor(anc__caron__comma,accent__default[anc__caron__comma]);
  tsu__default__anchor(anc__lower,accent__default[anc__lower]);
  tsu__default__anchor(anc__lower_connect,accent__default[anc__lower_connect]);
  tsu__default__anchor(anc__centre,tsu__xf.low_centre_accent);
enddef;

vardef  tsu__accent.clear_default_anchors =
  tsu__default__anchor(anc__upper,0);
  tsu__default__anchor(anc__grave,0);
  tsu__default__anchor(anc__acute,0);
  tsu__default__anchor(anc__wide,0);
  tsu__default__anchor(anc__tilde,0);
  tsu__default__anchor(anc__ring,0);
  tsu__default__anchor(anc__caron__comma,0);
  tsu__default__anchor(anc__lower,0);
  tsu__default__anchor(anc__lower_connect,0);
  tsu__default__anchor(anc__centre,0);
enddef;
```

accent.mp

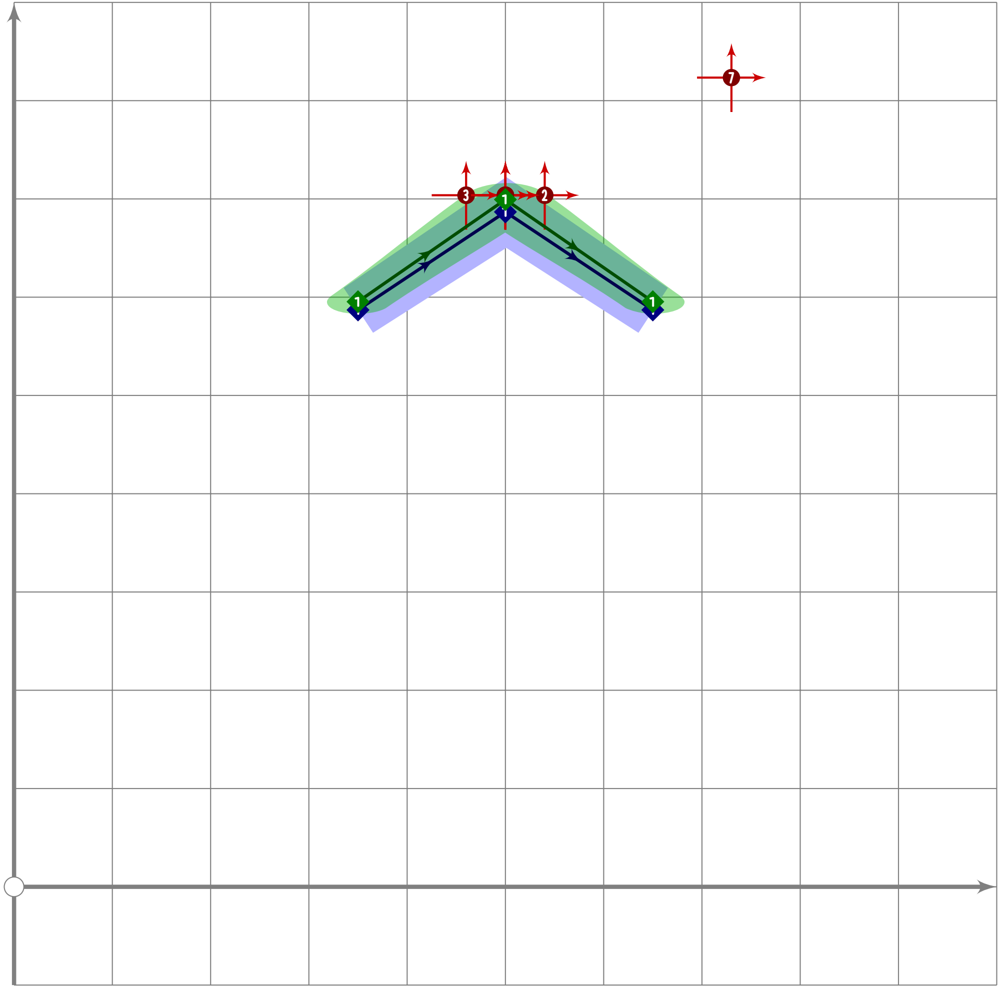
```
%
% Accents for Tsukurimashou
% Copyright (C) 2011, 2012 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(accent);

```



```

vardef tsu_accent.acute =
  push_anchor(-anc_acute,accent_default[anc_acute]);
  push_stroke(
    (500+1.1*tsu_punct_size,vmetric(0.95))--
      (500-0.9*tsu_punct_size,vmetric(0.78)),
    (2,2)--(1.6,1.6)--(1.3,1.3));
  replace_strokep(0)(insert_nodes(oldp)(0.5));
```

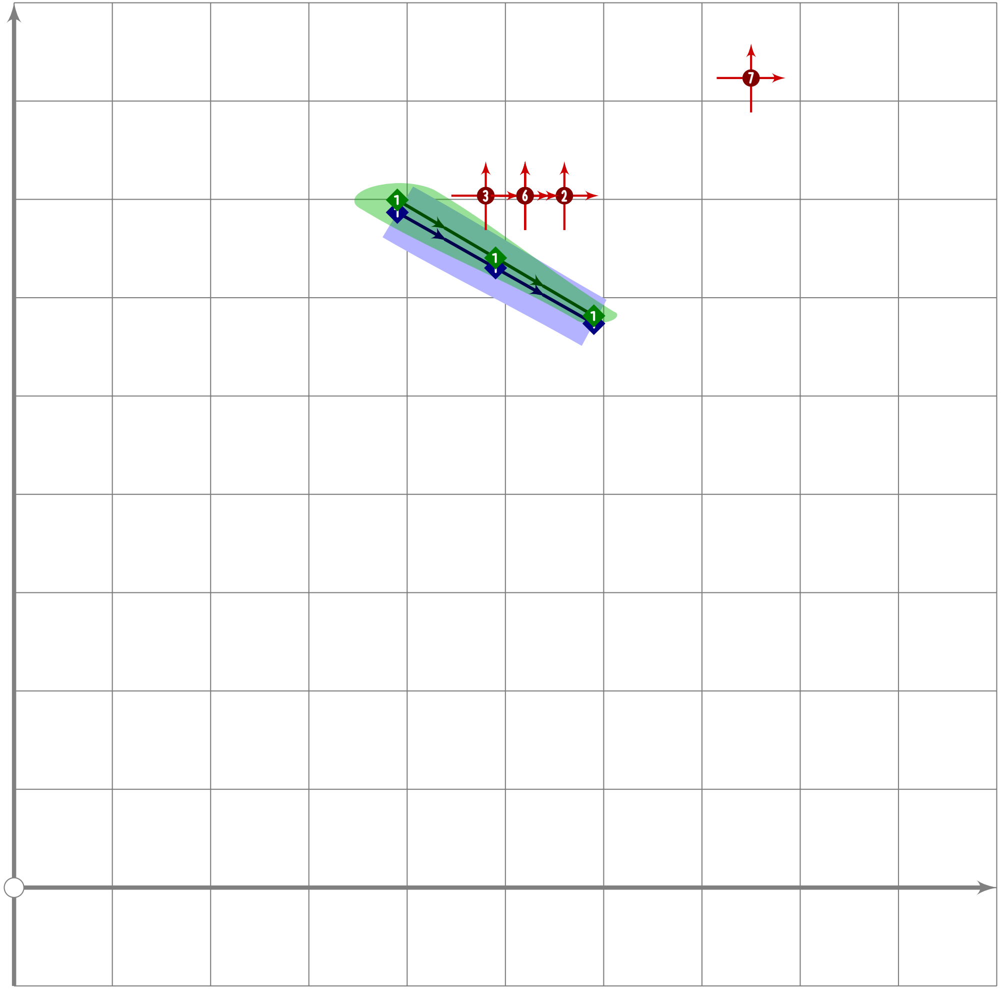
ACCE

```
  set_bosize(0,80);
  set_botip(0,1,1);
  push_anchor(anc_upper,accent_default[anc_upper] shifted (-20,150));
  push_anchor(anc_grave,accent_default[anc_grave] shifted (-20,150));
  push_anchor(anc_acute,accent_default[anc_acute] shifted (-20,150));
  push_anchor(anc_wide,accent_default[anc_wide] shifted (-20,150));
  push_anchor(anc_tilde,accent_default[anc_tilde] shifted (-20,150));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (-20,150));
  push_anchor(anc_caron_comma,
              accent_default[anc_caron_comma] shifted (-20,150));
enddef;

vardef tsu_accent.breve =
  push_anchor(-anc_wide,accent_default[anc_wide]);
  push_stroke((500-1.3*tsu_punct_size,vmetric(0.95)){down}..
      (500,vmetric(0.82))..
      {up}(500+1.3*tsu_punct_size,vmetric(0.95)),
    (1,1)--(1.9,1.9)--(1,1));
  push_anchor(anc_upper,accent_default[anc_upper] shifted (0,150));
  push_anchor(anc_grave,accent_default[anc_grave] shifted (0,150));
  push_anchor(anc_acute,accent_default[anc_acute] shifted (0,150));
  push_anchor(anc_wide,accent_default[anc_wide] shifted (0,150));
  push_anchor(anc_tilde,accent_default[anc_tilde] shifted (0,150));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (0,150));
  push_anchor(anc_caron_comma,
              accent_default[anc_caron_comma] shifted (0,150));
enddef;

vardef tsu_accent.caron =
  push_anchor(-anc_wide,accent_default[anc_wide]);
  push_stroke((500-1.5*tsu_punct_size,vmetric(0.95))--
    (500,vmetric(0.80))--
    (500+1.5*tsu_punct_size,vmetric(0.95)),
  (2,2)--(1.4,1.4)--(1.4,1.4));
  set_bosize(0,80);
  set_botip(0,1,1);
  push_anchor(anc_upper,accent_default[anc_upper] shifted (0,150));
  push_anchor(anc_grave,accent_default[anc_grave] shifted (0,150));
  push_anchor(anc_acute,accent_default[anc_acute] shifted (0,150));
  push_anchor(anc_wide,accent_default[anc_wide] shifted (0,150));
  push_anchor(anc_tilde,accent_default[anc_tilde] shifted (0,150));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (0,150));
  push_anchor(anc_caron_comma,
              accent_default[anc_caron_comma] shifted (0,150));
enddef;

vardef tsu_accent.caron_comma =
  push_anchor(-anc_caron_comma,accent_default[anc_caron_comma]);
  punct.make_comma((0,0) transformed accent_default[anc_caron_comma],0);
enddef;

vardef tsu_accent.cedilla =
  push_anchor(-anc_lower_connect,accent_default[anc_lower_connect]);
  push_stroke(
    ((0,0)--(-0.3,-1.8){curl 0.7}..(2.6,-2.5)..{curl 0.2}(-2.5,-3.0))
      scaled (0.5*tsu_punct_size) shifted (500,latin_wide_low_r),
    (1.4,1.4)--(1.4,1.4)--(1.7,1.7)--(1.3,1.3));
  set_bosize(0,80);
  set_botip(0,1,1);
enddef;
```



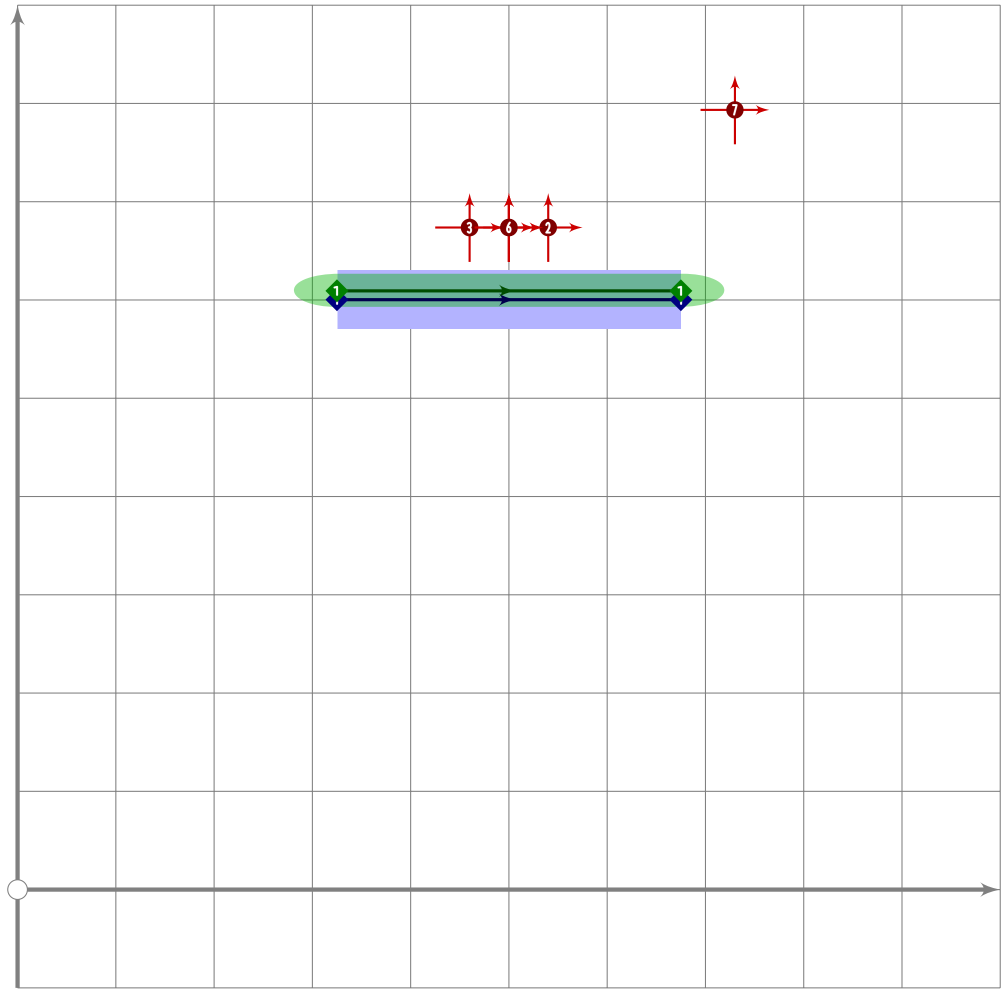
```

vardef tsu_accent.circumflex =
  push_anchor(-anc_wide,accent_default[anc_wide]);
  push_stroke((500-1.5*tsu_punct_size,vmetric(0.80))--
      (500,vmetric(0.95))--
      (500+1.5*tsu_punct_size,vmetric(0.80)),
    (1.6,1.6)--(2,2)--(1.6,1.6));
  set_bosize(0,80);
  set_botip(0,1,1);
  push_anchor(anc_upper,accent_default[anc_upper] shifted (0,150));
  push_anchor(anc_grave,accent_default[anc_grave] shifted (0,150));
  push_anchor(anc_acute,accent_default[anc_acute] shifted (0,150));
  push_anchor(anc_wide,accent_default[anc_wide] shifted (0,150));
  push_anchor(anc_tilde,accent_default[anc_tilde] shifted (0,150));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (0,150));
  push_anchor(anc_caron_comma,
              accent_default[anc_caron_comma] shifted (0,150));
enddef;

vardef tsu_accent.commabelow =
  push_anchor(-anc_lower,accent_default[anc_lower]);
  punct.make_comma((0,0) transformed accent_default[anc_lower],0);
enddef;

vardef tsu_accent.dotabove =
  push_anchor(-anc_upper,accent_default[anc_upper]);
  push_lcblob(fullcircle rotated 45 scaled (mbrush_width*1.72+50)
    shifted ((500,vmetric(0.88))
      transformed tsu_rescale_xform)
    transformed inverse tsu_rescale_xform);
  set_bokeepshape(0);
  push_anchor(anc_upper,accent_default[anc_upper] shifted (0,150));
  push_anchor(anc_grave,accent_default[anc_grave] shifted (0,150));
  push_anchor(anc_acute,accent_default[anc_acute] shifted (0,150));
  push_anchor(anc_wide,accent_default[anc_wide] shifted (0,150));
  push_anchor(anc_tilde,accent_default[anc_tilde] shifted (0,150));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (0,150));
  push_anchor(anc_caron_comma,
              accent_default[anc_caron_comma] shifted (0,150));
enddef;

vardef tsu_accent.dotbelow =
  push_anchor(-anc_lower,accent_default[anc_lower]);
  push_lcblob(fullcircle rotated 45 scaled (mbrush_width*1.72+50)
    shifted ((0,0) transformed accent_default[anc_lower]
      transformed tsu_rescale_xform)
    transformed inverse tsu_rescale_xform);
  set_bokeepshape(0);
enddef;

vardef tsu_accent.dotcentred =
  push_anchor(-anc_centre,accent_default[anc_centre]);
  push_lcblob(fullcircle rotated 45 scaled (mbrush_width*1.72+50)
    shifted ((0,0) transformed accent_default[anc_centre]
      transformed tsu_rescale_xform)
    transformed inverse tsu_rescale_xform);
  set_bokeepshape(0);
```

```
enddef;
```



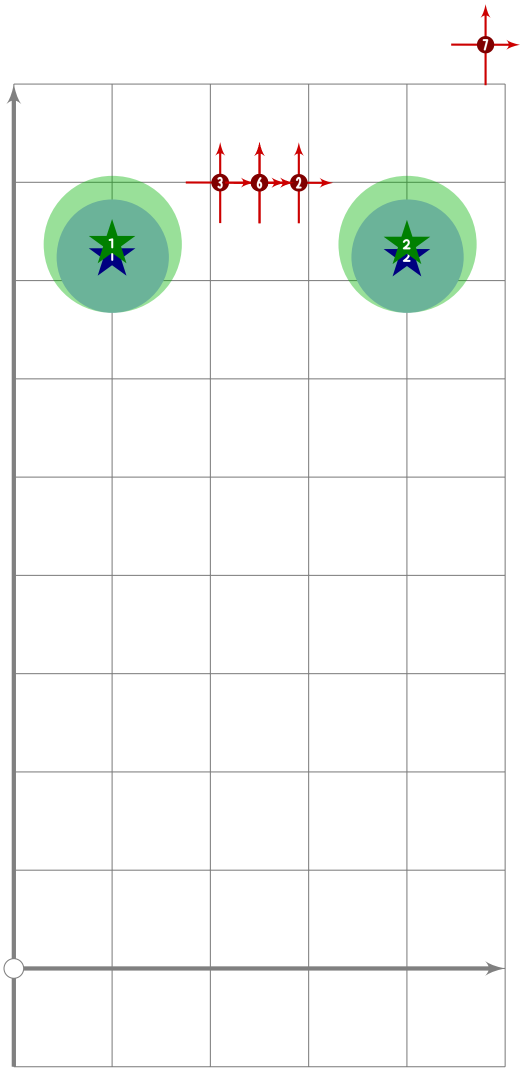
```

vardef tsu_accent.grave =
  push_anchor(-anc_grave,accent_default[anc_grave]);
  push_stroke((500-1.1*tsu_punct_size,vmetric(0.95))--
      (500+0.9*tsu_punct_size,vmetric(0.78)),
    (2,2)--(1.6,1.6)--(1.3,1.3));
  replace_strokep(0)(insert_nodes(oldp)(0.5));
  set_bosize(0,80);
  set_botip(0,1,1);
  push_anchor(anc_upper,accent_default[anc_upper] shifted (20,150));
  push_anchor(anc_grave,accent_default[anc_grave] shifted (20,150));
  push_anchor(anc_acute,accent_default[anc_acute] shifted (20,150));
  push_anchor(anc_wide,accent_default[anc_wide] shifted (20,150));
  push_anchor(anc_tilde,accent_default[anc_tilde] shifted (20,150));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (20,150));
  push_anchor(anc_caron_comma,
              accent_default[anc_caron_comma] shifted (20,150));
enddef;

vardef tsu_accent.heavy_metal_umlaut =
  push_anchor(-anc_wide,accent_default[anc_wide]);
  push_lcblob((up--left--down--right--cycle) scaled (mbrush_width*1.5+30)
    shifted ((500-1.5*tsu_punct_size,vmetric(0.88))
      transformed tsu_rescale_xform)
    transformed inverse tsu_rescale_xform);
  push_lcblob((up--left--down--right--cycle) scaled (mbrush_width*1.5+30)
    shifted ((500+1.5*tsu_punct_size,vmetric(0.88))
      transformed tsu_rescale_xform)
    transformed inverse tsu_rescale_xform);
  set_bokeepshape(-1);
  set_bokeepshape(0);
  push_anchor(anc_upper,accent_default[anc_upper] shifted (0,220));
  push_anchor(anc_grave,accent_default[anc_grave] shifted (0,220));
  push_anchor(anc_acute,accent_default[anc_acute] shifted (0,220));
  push_anchor(anc_wide,accent_default[anc_wide] shifted (0,220));
  push_anchor(anc_tilde,accent_default[anc_tilde] shifted (0,220));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (0,220));
  push_anchor(anc_caron_comma,
              accent_default[anc_caron_comma] shifted (0,220));
enddef;

vardef tsu_accent.hungarian_umlaut =
  push_anchor(-anc_wide,accent_default[anc_wide]);
  push_stroke((500+1.7*tsu_punct_size,vmetric(0.95))--
      (500+0.7*tsu_punct_size,vmetric(0.78)),
    (2,2)--(1.6,1.6)--(1.3,1.3));
  replace_strokep(0)(insert_nodes(oldp)(0.5));
  set_bosize(0,80);
  set_botip(0,1,1);

  push_stroke((500-0.4*tsu_punct_size,vmetric(0.95))--
      (500-1.4*tsu_punct_size,vmetric(0.78)),
    (2,2)--(1.6,1.6)--(1.3,1.3));
  replace_strokep(0)(insert_nodes(oldp)(0.5));
  set_bosize(0,80);
  set_botip(0,1,1);
  push_anchor(anc_upper,accent_default[anc_upper] shifted (0,180));
  push_anchor(anc_grave,accent_default[anc_grave] shifted (0,180));
  push_anchor(anc_acute,accent_default[anc_acute] shifted (0,180));
  push_anchor(anc_wide,accent_default[anc_wide] shifted (0,180));
  push_anchor(anc_tilde,accent_default[anc_tilde] shifted (0,180));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (0,180));
  push_anchor(anc_caron_comma,
              accent_default[anc_caron_comma] shifted (0,180));
enddef;
```

U+FFE3
tsuku.uniFFE3



```

vardef tsu_accent.macron =
  push_anchor(-anc_wide,accent_default[anc_wide]);
  push_stroke((500-1.75*tsu_punct_size,vmetric(0.82))--
      (500+1.75*tsu_punct_size,vmetric(0.82)),
    (2,2)--(2,2));
  set_bosize(0,80);
  push_anchor(anc_upper,accent_default[anc_upper] shifted (0,120));
  push_anchor(anc_grave,accent_default[anc_grave] shifted (0,120));
  push_anchor(anc_acute,accent_default[anc_acute] shifted (0,120));
  push_anchor(anc_wide,accent_default[anc_wide] shifted (0,120));
  push_anchor(anc_tilde,accent_default[anc_tilde] shifted (0,120));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (0,120));
  push_anchor(anc_caron_comma,
             accent_default[anc_caron_comma] shifted (0,120));
enddef;

vardef tsu_accent.ringabove =
  push_anchor(-anc_ring,accent_default[anc_ring]);
  push_lcblob(fullcircle rotated 45
    scaled (2*tsu_punct_size+20*tsu_brush_max)
    shifted ((500,vmetric(0.83-0.03*mincho)-10)
      transformed tsu_rescale_xform)
    transformed inverse tsu_rescale_xform);
  set_bokeepshape(0);
  if 2*tsu_punct_size-110*tsu_brush_max>10:
    push_lcblob(reverse fullcircle rotated 45
      scaled (2*tsu_punct_size-110*tsu_brush_max)
      shifted ((500,vmetric(0.83-0.03*mincho)-10)
        transformed tsu_rescale_xform)
      transformed inverse tsu_rescale_xform);
    set_bokeepshape(0);
  fi;
  push_anchor(anc_upper,accent_default[anc_upper] shifted (0,170));
  push_anchor(anc_grave,accent_default[anc_grave] shifted (0,170));
  push_anchor(anc_acute,accent_default[anc_acute] shifted (0,170));
  push_anchor(anc_wide,accent_default[anc_wide] shifted (0,170));
  push_anchor(anc_tilde,accent_default[anc_tilde] shifted (0,170));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (0,170));
  push_anchor(anc_caron_comma,
              accent_default[anc_caron_comma] shifted (0,170));
enddef;

vardef tsu_accent.slash =
  push_anchor(-anc_centre,accent_default[anc_centre]);
  push_stroke(((-100,-130)--(100,130))
    transformed accent_default[anc_centre],(2,2)--(2,2));
  set_bosize(0,86);
enddef;

vardef tsu_accent.tilde =
  push_anchor(-anc_tilde,accent_default[anc_tilde]);
  push_stroke(
    ((-3.5,-0.5){curl 0}..(-1.4,1)..(0,0)..(1.4,-1)..{curl 0}(3.5,0.5))
      rotated 5 xyscaled (0.7*tsu_punct_size,0.5*tsu_punct_size)
      shifted (500,vmetric(0.85)),
    (0.7,2.7)--(1.7,1.7)--(1.7,1.7)--(1.7,1.7)--(1.7,1.7)--
      (1.7,1.7)--(0.7,2.7));
  replace_strokep(0)(insert_nodes(oldp)(0.5,3.5));
  set_bosize(0,80);
  push_anchor(anc_upper,accent_default[anc_upper] shifted (0,150));
  push_anchor(anc_grave,accent_default[anc_grave] shifted (0,150));
  push_anchor(anc_acute,accent_default[anc_acute] shifted (0,150));
  push_anchor(anc_wide,accent_default[anc_wide] shifted (0,150));
  push_anchor(anc_tilde,accent_default[anc_tilde] shifted (0,150));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (0,150));
  push_anchor(anc_caron_comma,
              accent_default[anc_caron_comma] shifted (0,150));
enddef;
```

U+00A8
tsuku.dieresis



```

vardef tsu_accent.umlaut =
  push_anchor(-anc_wide,accent_default[anc_wide]);
  push_lcblob(fullcircle rotated 45 scaled (mbrush_width*1.72+50)
    shifted ((500-1.5*tsu_punct_size,vmetric(0.88))
      transformed tsu_rescale_xform)
    transformed inverse tsu_rescale_xform);
  push_lcblob(fullcircle rotated 45 scaled (mbrush_width*1.72+50)
    shifted ((500+1.5*tsu_punct_size,vmetric(0.88))
      transformed tsu_rescale_xform)
    transformed inverse tsu_rescale_xform);
  set_bokeepshape(-1);
  set_bokeepshape(0);
  push_anchor(anc_upper,accent_default[anc_upper] shifted (0,150));
  push_anchor(anc_grave,accent_default[anc_grave] shifted (0,150));
  push_anchor(anc_acute,accent_default[anc_acute] shifted (0,150));
  push_anchor(anc_wide,accent_default[anc_wide] shifted (0,150));
  push_anchor(anc_tilde,accent_default[anc_tilde] shifted (0,150));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (0,150));
  push_anchor(anc_caron_comma,
              accent_default[anc_caron_comma] shifted (0,150));
enddef;

────────────────────────────────────────────────────────────────────────────────────────

vardef tsu_accent.capital(text curves) =
  tsu_xform(tsu_xf.accentedcap)
    (curves;tsu_accent.shift_anchors(true)((0,0)));
enddef;

vardef tsu_accent.apply(text basecurves)(text markcurves) =
  begingroup;
    save xsp,ysp,bmi,mbi,bmt,killflag;
    numeric xsp,ysp,bmi,mbi;
    transform bmt;
    boolean killflag;
    basecurves;
    xsp:=sp;
    markcurves;
    killflag:=false;
    for i:=sp-1 downto xsp:
      if obstacktype[i]=otanchor:
        if obstackn[i]<0:
          mbi:=i;
          killflag:=true;
        fi;
      fi;
      exitif killflag;
    endfor;
    if known mbi:
      if known accent_default[-obstackn[mbi]]:
        bmt:=accent_default[-obstackn[mbi]];
      else:
        bmt:=accent_default[anchor_parent[-obstackn[mbi]]];
      fi;
      killflag:=false;
      for i:=xsp-1 downto 1:
        if obstacktype[i]=otanchor:
          if obstackn[i]=-obstackn[mbi]:
            bmi:=i;
            bmt:=obstackt[i];
            killflag:=true;
          fi;
        fi;
        exitif killflag;
      endfor;
      if (not killflag) and (known anchor_parent[-obstackn[mbi]]):
        for i:=xsp-1 downto 1:
          if obstacktype[i]=otanchor:
            if obstackn[i]=anchor_parent[-obstackn[mbi]]:
              bmi:=i;
              bmt:=obstackt[i];
              killflag:=true;
            fi;
          fi;
          exitif killflag;
        endfor;
      fi;
      obstacktype[mbi]:=otnull;
      if known bmi:
        obstacktype[bmi]:=otnull;
      fi;
      ysp:=sp;
      sp:=xsp;
      tsu_xform((inverse obstackt[mbi]) transformed bmt)(sp:=ysp);
    fi;
  endgroup;
enddef;
```

bcircle.mp

```
%
% Bounding circle algorithm of E. Welzl
% Copyright (C) 2011 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(bcircle);

% swap points in pts[] array
vardef swap_pts(expr a,b)=
  pair tmppt;
  tmppt:=pts[a];
  pts[a]:=pts[b];
  pts[b]:=tmppt;
enddef;

% compute bounding circle on up to three points
vardef bcircle.basis(expr rstart,rend) =
  if rend<=rstart+1:
    identity
  else:
    begingroup
      save x,y,myt;
      numeric x[],y[];
      transform myt;
      z1=pts[rstart];
      z2=pts[rstart+1];
      xypart myt=0;
      yxpart myt=0;
      if rend=rstart+2:
        z3=(z1+z2)/2;
        (0,0) transformed myt=(z1+z2)/2;
        xxpart myt=yypart myt=abs(z1-z3);
      else:
        z3=pts[rstart+2];
        z4=(z1+z2)/2;
        z5=(z1+z3)/2;
        z6=z4+whatever*((z2-z1) rotated 90);
        z6=z5+whatever*((z3-z1) rotated 90);
        (0,0) transformed myt=z6;
        xxpart myt=yypart myt=abs(z1-z6);
      fi;
      myt
    endgroup
  fi
enddef;
```

```

% recursion to compute bounding circle.
% Input point sets are in pts[] array, arguments are indices into it
vardef bcircle.internal(expr pstart,rstart,rend) =
  if (pstart=rstart) or (rend-rstart=3):
    bcircle.basis(rstart,rend)
  else:
    begingroup
      transform d;
      pind:=floor ((rstart-pstart)*uniformdeviate 1)+pstart;
      swap__pts(pstart,pind);
      d=bcircle.internal(pstart+1,rstart,rend);
      pair xpt;
      xpt transformed d=pts[pstart];
      if abs(xpt)>1:
        swap__pts(pstart,(rstart-1));
        d:=bcircle.internal(pstart,rstart-1,rend);
      fi;
      d
    endgroup
  fi
enddef;

% wrapper for bounding circle algorithm - compute bcircle of points
vardef bcircle.points(text txt) =
  begingroup
    save d,tmppt,pind,xpt,pts,pcnt;
    pcnt:=0;
    for myp=txt:
      pts[pcnt]:=myp;
      pcnt:=pcnt+1;
    endfor;
    bcircle.internal(0,pcnt,pcnt)
  endgroup
enddef;

% wrapper for bounding circle algorithm - compute bcircle of paths
vardef bcircle.paths(text txt) =
  begingroup
    save d,tmppt,pind,xpt,pts,pcnt;
    pcnt:=0;
    for myp=txt:
      for i=0 step 0.1 until length myp:
        pts[pcnt]:=point i of myp;
        pcnt:=pcnt+1;
      endfor
    endfor;
    bcircle.internal(0,pcnt,pcnt)
```

```
   endgroup
enddef;
```

buildkanji.mp

```
%
% Build a kanji character by assembling parts
% Copyright (C) 2011, 2012, 2013 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(buildkanji);

```

```

% a beret is a tilted hat
vardef build_kanji.add_beret(text curves) =
  perl_structure:=perl_structure
    &"['build_kanji.add_beret",eids.u2FF1.u4E3F._1",";
  begingroup
    save osp;
    numeric osp;
    osp:=sp;
    curves;
    save i,lox,hix,toppt,myxf;
    lox:=infinity;
    hix:=-infinity;
    pair toppt;
    toppt:=(500,-infinity);
    i:=0;
    forever:
      exitif find_stroke(i)<osp;
      if xpart llcorner get_strokep(i)<lox:
        lox:=xpart llcorner get_strokep(i);
      fi;
      if xpart urcorner get_strokep(i)>hix:
        hix:=xpart urcorner get_strokep(i);
      fi;
      if ypart point 0 of get_strokep(i)>ypart toppt:
        toppt:=point 0 of get_strokep(i);
      fi;
      if ypart point infinity of get_strokep(i)>ypart toppt:
        toppt:=point infinity of get_strokep(i);
      fi;
      i:=i-1;
    endfor;
    i:=0;
    forever:
      exitif find_stroke(i)<osp;
      if point 0 of get_strokep(i)=toppt:
        set_boserif(i,0,whatever);
      fi;
      if point infinity of get_strokep(i)=toppt:
        set_boserif(i,length get_strokep(i),whatever);
      fi;
      i:=i-1;
    endfor;
    transform myxf;
    (-500,810) transformed myxf=(lox,(ypart toppt)+30);
    (500,900) transformed myxf=(hix,900);
    (0,780) transformed myxf=toppt;
    push_stroke(
      ((-400,750)..(0,780)..tension 1.1..(360,840)) transformed myxf,
      (1.1,1.1)--(1.6,1.6)--(2.0,2.0));
  endgroup;
  perl_structure:=perl_structure&"],";
enddef;

vardef build_kanji.add_jtail(expr idx) =
  replace_strokep(idx)(oldp--(xpart point infinity of oldp,30){down}..
    {curl 0.2}((xpart point infinity of oldp)-150,0));
  replace_strokep(idx)(insert_nodes(oldp)((length oldp)-0.5));
  replace_strokeq(idx)(oldq--(1.5,1.5)--(1.4,1.4)--(1.2,1.2));
enddef;

% hook used by extend_ltail_enclose
numeric last_ltail;

vardef build_kanji.add_ltail(expr idx) =
  begingroup
    save x,y;
    numeric x[],y[];
    z1=point infinity of get_strokep(idx);
    last_ltail:=find_stroke(idx);
    x2=x1;
    y2=80-40*mincho;
    x3=0.4[x2,800];
    y3=mincho[0,-20];
    replace_strokep(idx)(interpath(mincho,
      oldp--z2{down}...{right}z3..(750,0)..(810,30)..
        tension 2..(850,230),
      oldp--z2{down}...{right}z3..(850,0){curl 0.2}..(810,60)..
        tension 2..(810,170)));
    replace_strokeq(idx)(oldq--(1.6,1.6)--(1.75,1.75)--(1.9,1.9)
      --(1.3,1.3)--(0.8,0.8));
  endgroup;
enddef;

vardef build_kanji.attach_fishhook(expr scaleamt)(text curves) =
perl_structure:=
  perl_structure&"['build_kanji.attach_fishhook','eids.u2FF1.u2E88._2',[";
begingroup
  save osp;
  numeric osp;
  osp:=sp;
  tsu_xform(identity shifted (0,50) yscaled scaleamt shifted (0,-50))
    (curves);
  perl_structure:=perl_structure&"],";
  save i,j,x,y,pp,myxf;
  path pp;
  transform myxf;
  z1=z2=(-infinity,-infinity);
  i:=0;
  forever:
    exitif find_stroke(i)<osp;
    pp:=get_strokep(i) rotated -45;
    for j=0 upto length pp:
      x3:=xpart point j of pp;
      y3:=ypart point j of pp;
      if x3>x1:
        x1:=x3;
        y1:=y3;
      fi;
      if y3>y2:
        x2:=x3;
        y2:=y3;
      fi;
    endfor;
    i:=i-1;
  endfor;
  (0,0) transformed myxf=(z2 rotated 45);
  (1.8,0) transformed myxf=(z1 rotated 45);
  xypart myxf=0;
  ypart ((0,1) transformed myxf)=830;
  push_stroke(
    ((0.5,0.9)..tension 1.2..(0,0)..(-0.5,-0.5)) transformed myxf,
    (1.7,1.7)--(1.3,1.3)--(1,1));
  set_boserif(0,0,10);
  push_stroke(
    ((0.4,0.65)--(1.3,0.65)..tension 1.2..(1.14,0.3)..(0.88,0))
       transformed myxf,
    (1.6,1.6)--(1.6,1.6)--(1.4,1.4)--(1,1));
  set_boserif(0,1,4);
  set_botip(0,1,0);
endgroup;
perl_structure:=perl_structure&"],";
enddef;
```

```

vardef build_kanji.attach_tick(expr newtop)(text curves) =
  perl_structure:=
    perl_structure&”[’build_kanji.attach_fishhook’,”eids.u2FF1.u31D2._1’,[”;
  begingroup
    save atosp,atnsp;
    numeric atosp,atnsp;
    atosp:=sp;
    curves;
    perl_structure:=perl_structure&”],”;
    atnsp:=sp;
    save i,thisy,maxy;
    maxy:=-infinity;
    i:=0;
    forever:
      exitif find_stroke(i)<atosp;
      if (xpart (get_strokep(i) intersectiontimes
            ((400,900)–(500,200)–(600,900))))>=0:
        thisy:=ypart urcorner get_strokep(i);
        if thisy>maxy:
          maxy:=thisy;
        fi;
      fi;
      i:=i-1;
    endfor;
    sp:=atosp;
    tsu_xform(identity yscaled (newtop/maxy))(sp:=atnsp);
    push_stroke((500,190+newtop)–(440,newtop),(1.7,1.7)–(1,1));
    set_boserif(0,0,10);
  endgroup;
  perl_structure:=perl_structure&”],”;
enddef;

vardef hook.box_bottom(expr sides_i,bottom_i) =
  if (obstacktype[sides_i]=otstroke) and (obstacktype[bottom_i]=otstroke):
    if (3=length obstackp[sides_i]) and (1=length obstackp[bottom_i]):
      begingroup;
        save x,y,p,e;
        numeric x[],y[],e[];
        path p[];

        p1=obstackp[sides_i];
        p2=obstackp[bottom_i];

        z1=-(direction 0 of p1)/abs(direction 0 of p1);
        z2=(direction 0.5 of p2)/abs(direction 0.5 of p2);
        z3=(direction 3 of p1)/abs(direction 3 of p1);

        if (abs(z1 dotprod z2)<0.1) and (abs(z3 dotprod z2)<0.1):
          point 0 of p1=z4+e1*z1;
          z4=(point 0 of p2)+whatever*z2;
          point 3 of p1=z5+e2*z3;
          z5=(point 1 of p2)+whatever*z2;
          e3=obstackna.bosize[bottom_i]*tsu_brush_max*tsu_brush_shape*0.5;

          if e1<e3:
            p1:=(z4+e3*z1)--(subpath (1,3) of p1);
          fi;
          if e2<e3:
            p1:=(subpath (0,2) of p1)--(z5+e3*z3);
          fi;
          obstackp[sides_i]:=p1;
        fi;
      endgroup;
    fi;
  fi;
enddef;

vardef build_kanji.box(expr ul,lr) =
  perl_structure:=perl_structure&"['build_kanji.box',";
  begingroup
    save boxext;
    if (ypart (ul-lr))>500:
      boxext:=-100/(ypart (ul-lr));
    else:
      boxext:=-0.2;
    fi;
    if (boxext)[ypart lr,ypart ul]<-60:
      boxext:=(-60+ypart lr)/(ypart ul-ypart lr);
    fi;
    push_stroke((xpart ul,(boxext)[ypart lr,ypart ul])--ul--
        (xpart lr,ypart ul)--(xpart lr,(boxext)[ypart lr,ypart ul]),
      (1.5,1.5)--(1.7,1.7)--(1.7,1.7)--(1.5,1.5));
  endgroup;
  set_botip(0,1,1);
  set_botip(0,2,1);
  set_boserif(0,1,4);
  set_boserif(0,2,4);
  push_stroke((xpart ul,ypart lr)--lr,
    (1.5,1.5)--(1.5,1.5));
  push_hook(hsmain_render,
    "hook.box_bottom("&(decimal find_stroke(-1))&","
                     &(decimal find_stroke(0))&");" );
  perl_structure:=perl_structure&"],";
enddef;
```

```
vardef build_kanji.cliff_enclose(text contents) =
  push_pbox_toexpand("build_kanji.cliff_enclose");
  perl_structure:=perl_structure&"'eids.u2FF8.u5382.1_',";
  push_stroke((50,-50)..(120,100)..(160,300)..tension 1.2..(180,760)
      --(850,760),
    (1,1)--(1.3,1.3)--(1.5,1.5)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,3,1);
  begingroup
    save t;
    transform t;
    (50,-50) transformed t=(230,-50);
    (500,-50) transformed t=(560,-50);
    (500,850) transformed t=(560,730);
    tsu_xform(t)(contents);
  endgroup;
  expand_pbox;
enddef;

vardef build_kanji.cup(expr ul,lr) =
  push_stroke(ul--(xpart ul,(-0.2)[ypart lr,ypart ul]),
    (1.6,1.6)--(1.4,1.4));
  set_boserif(0,0,10);
  push_stroke((xpart lr,ypart ul)--(xpart lr,(-0.2)[ypart lr,ypart ul]),
    (1.6,1.6)--(1.4,1.4));
  set_boserif(0,0,10);
  push_stroke((xpart ul,ypart lr)--lr,
    (1.5,1.5)--(1.5,1.5));
  perl_structure:=perl_structure&"'build_kanji.cup',";
enddef;

vardef build_kanji.dotcliff_enclose(text contents) =
  push_pbox_toexpand("build_kanji.dotcliff_enclose");
  perl_structure:=perl_structure&"'eids.u2FF8.u5E7F.2_',";
  push_stroke((550,810)--(550,660),
    (1.6,1.6)--(1.5,1.5));
  set_boserif(0,0,10);
  push_stroke(
    (50,-50)..(120,100)..(160,300)..tension 1.2..(180,660)--(850,660),
    (1,1)--(1.3,1.3)--(1.5,1.5)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,3,1);
  begingroup
    save t;
    transform t;
    (50,-50) transformed t=(230,-50);
    (500,-50) transformed t=(560,-50);
    (500,850) transformed t=(560,630);
    tsu_xform(t)(contents);
  endgroup;
  expand_pbox;
enddef;

vardef build_kanji.flag_enclose(expr xs,ys)(text contents) =
  push_pbox_toexpand("build_kanji.dotcliff_enclose");
  perl_structure:=perl_structure&"'eids.u2FF8.u5C38.1_',";
  string flag_enclose.ps;
  flag_enclose.ps:=perl_structure;
  build_kanji.lr(460,60)
    (kanji.grtwo.direction;)
    (perl_structure:=flag_enclose.ps;
     push_stroke((300,810)..tension 1.2..(220,600)..(110,480),
       (1.6,1.6)--(1.4,1.4)--(1,1));
     set_boserif(0,0,10);
     push_stroke((100,680)--(780,680),(1.6,1.6)--(1.6,1.6));
     set_boserif(0,1,9);
     replace_strokep(0)(subpath
       (0.01+xpart (oldp intersectiontimes get_strokep(-1)),1) of oldp);
     tsu_xform(identity shifted (-500,0) xyscaled (xs,ys)
         shifted (500,-20))
       (contents);
     flag_enclose.ps:=perl_structure);
  expand_pbox;
  perl_structure:=flag_enclose.ps&"],";
enddef;

vardef build_kanji.gate_enclose(text contents) =
  perl_structure:=perl_structure&"['build_kanji.gate_enclose',";
  perl_structure:=perl_structure&"'eids.u2FF5.u9580.1_',";
  kanji.grtwo.gate;
  begingroup
    transform xf;
    (50,-50) transformed xf=(220,40);
    (950,850) transformed xf=(780,420);
    xypart xf=yxpart xf=0;
    tsu_xform(xf)(contents);
  endgroup;
  perl_structure:=perl_structure&"],";
enddef;

vardef build_kanji.harmonic(expr gap,sval,lspread)(text curves) =
  perl_structure:=perl_structure
    &"['build_kanji.harmonic','eids.u2FF1.u003F._1',[";
  begingroup
    save myxf;
    transform myxf;
    (50,-50) transformed myxf=(50,-50);
    (950,-50) transformed myxf=(950,-50);
    (50,850) transformed myxf=(50,850-gap);
    tsu_xform(myxf)(curves);
    perl_structure:=perl_structure&”],[”;
    save hsp;
    hsp:=sp;
    tsu_xform(myxf)(build_kanji.spread_legs(lspread)(curves));
    save i,tp,toclip,nadded;
    numeric i,toclip,nadded;
    path tp;
    i:=0;
    toclip:=0;
    nadded:=0;
    forever:
      tp:=get_strokep(i);
      if (ypart ulcorner tp<450)
        or (abs(ypart (direction 0 of tp)/abs(direction 0 of tp))>0.95)
        or (abs(ypart (direction infinity of tp)/
          abs(direction infinity of tp))>0.95):
        set_bosize(i,0);
        toclip:=toclip+1;
      else:
        replace_strokep(i)
          (tp shifted -(0.5[ulcorner tp,lrcorner tp])
              scaled sval
              shifted ((0,gap)+0.5[ulcorner tp,lrcorner tp]));
        set_bosize(i)(get_bosize(i)*sqrt(sval));
        toclip:=toclip+2;
        nadded:=nadded+1;
      fi;
      i:=i-1;
      exitif find_stroke(i)<hsp;
    endfor;
  endgroup;
  perl_structure:=perl_structure&”]],”;
enddef;

vardef build_kanji.lcr(expr splitpointa,overlapa)(expr splitpointb,overlapb)
  (text leftstuff)(text centrestuff)(text rightstuff) =
  perl_structure:=perl_structure
    &”[’build_kanji.lcr’,’eids.u2ff2._2.1_1.2_’,[”;
  begingroup
    save t;
    transform t[];
    yypart t1=yypart t2=yypart t3=1;
    ypart t1=yxpart t1=xypart t1=0;
    ypart t2=yxpart t2=xypart t2=0;
    ypart t3=yxpart t3=xypart t3=0;
    (50,0) transformed t1=(50,0);
    (950,0) transformed t1=(splitpointa+overlapa/2,0);
    (50,0) transformed t2=(splitpointa-overlapa/2,0);
    (950,0) transformed t2=(splitpointb+overlapb/2,0);
    (50,0) transformed t3=(splitpointb-overlapb/2,0);
    (950,0) transformed t3=(950,0);
    tsu_xform(t1)(leftstuff);
    perl_structure:=perl_structure&”],[”;
    tsu_xform(t2)(centrestuff);
    perl_structure:=perl_structure&”],[”;
    tsu_xform(t3)(rightstuff);
  endgroup;
  perl_structure:=perl_structure&”]],”;
enddef;

vardef build_kanji.lean_to(expr lr) =
  push_stroke((120,620)--(880,620),(1.6,1.6)--(1.6,1.6));
  set_boserif(0,1,9);
  begingroup
    save ltxf;
    transform ltxf;
    xypart ltxf=yxpart ltxf=0;
    (60,800) transformed ltxf=(60,800);
    (360,-30) transformed ltxf=lr;
    push_stroke(
      ((360,800)..tension 1.2..(270,300)..(60,-30)) transformed ltxf,
      (1.6,1.6)--(1.4,1.4)--(0.9,0.9));
  endgroup;
  set_boserif(0,0,10);
enddef;

vardef build_kanji.level(text curves) =
  begingroup
    save xsp;
    xsp:=sp;
    curves;
    save lsum,denom,i;
    lsum:=0;
    denom:=0;
    i:=0;
    forever:
      exitif find_stroke(i)<xsp;
      if unknown get_bosize(i):
        set_bosize(i,100);
      fi;
      if (get_bosize(i)>0):
        lsum:=lsum+mlog(get_bosize(i));
        denom:=denom+1;
      fi;
      i:=i-1;
    endfor;
    i:=0;
    forever:
      exitif find_stroke(i)<xsp;
      if get_bosize(i)>0:
        set_bosize(i,mexp(lsum/denom));
      fi;
      i:=i-1;
    endfor;
  endgroup;
enddef;

vardef build_kanji.lstransform(expr thresh,dist,mypt) =
  if ypart mypt>thresh:
    mypt
  else:
    (xpart mypt,
      (ypart mypt)*((thresh-dist)/thresh)
     +(xpart mypt/1000)[(dist/thresh)*ypart mypt,dist])
  fi
enddef;

vardef build_kanji.lift_skirt(expr thresh,dist)(text curves) =
  begingroup
    save osp;
    numeric osp;
    osp:=sp;
    curves;
    save i,boti,x,y;
    numeric x[],y[];
    i:=0;
    boti:=whatever;
    y0:=1000;
    forever:
      exitif find_stroke(i)<osp;
      if (ypart llcorner get_strokep(i))=(ypart urcorner get_strokep(i)):
        if (unknown boti) or (ypart llcorner get_strokep(i)<y0):
          y0:=ypart llcorner get_strokep(i);
          boti:=i;
        fi;
      fi;
      replace_strokep(i)(
        for j=0 upto length oldp-1:
          build_kanji.lstransform(thresh,dist)(point j of oldp)
          ..controls build_kanji.lstransform(thresh,dist)(postcontrol j of oldp)
          and build_kanji.lstransform(thresh,dist)(precontrol j+1 of oldp)..
        endfor
        if cycle oldp:
          cycle
        else:
          build_kanji.lstransform(thresh,dist)(point infinity of oldp)
        fi);
      i:=i-1;
    endfor;
    if (known boti) and (y0<=thresh):
      replace_strokep(boti)
        (((0,7)+point 0 of oldp)..tension 1.2..((0,-5)+point 0.5 of oldp)..
         ((0,10)+point 1 of oldp));
      replace_strokeq(boti)((1.7,1.7)--(1.5,1.5)--(1,1));
      set_boserif(boti,1,whatever);
    fi;
  endgroup;
enddef;

% note special calling convention - extra boolean for inclusion of ”tick”
% seen in some Japanese and Korean characters
vardef build_kanji.long_stride_enclose(expr do_tick)(text contents) =
  push_pbox_toexpand(”build_kanji.long_stride_enclose”);
  perl_structure:=perl_structure&”’eids.u2FFA.u5EF4._3’”;
  begingroup
    save myxf;
    transform myxf;
    (50,850) transformed myxf=(350,810);
    (50,-50) transformed myxf=(350,80);
    (950,-50) transformed myxf=(950,80);
    tsu_xform(myxf)(contents);
    save x,y;
    numeric x[],y[];
    x1=80;
    x2=260;
    x4=120;
    y1=y2=780;
    y4=460;
    z3=0.4[z4,z2]+mincho*(30,0);
    z5=(0.3-0.5*mincho)[z4,z2];
    push_stroke(z1--z2..tension 1.2..z3..z5,
      (1.6,1.6)--(1.6,1.6)--(1.6-0.1*mincho,1.6-0.1*mincho)--
        (1.6-0.2*mincho,1.6-0.2*mincho));
    set_boserif(0,1,4);
    set_botip(0,1,0);
    x7=0;
    x9=300;
    x10=220;
    x11=100;
    y7=y9=y4;
    y10=80;
    y11=-70;
    z6=point 2.2 of get_strokep(0);
    z8=(z7--z9) intersectionpoint (get_strokep(0)--((-1)[z4,z2]));
    push_stroke(z6--z8--z9..tension 1.2..z10..z11,
      (1.6-0.5*mincho,1.6-0.5*mincho)--(1.5-0.1*mincho,1.5-0.1*mincho)--
      (1.6,1.6)--(1.4,1.4)--(1,1));
    set_boserif(0,2,4);
    set_botip(0,1,1);
    set_botip(0,2,0);
    if do_tick:
      push_stroke((150,330)..(120,290)..(60,220),
        (1.2,1.2)--(1.1,1.1)--(1.6,1.6));
    fi;
    push_stroke((130,300)..(390,30)..tension 1.6..(900,-50),
      (1,1)--(1.6,1.6)--(1.9,1.9));
  endgroup;
  expand_pbox;
enddef;

vardef build_kanji.lr(expr splitpoint,overlap)
  (text leftstuff)(text rightstuff) =
  perl_structure:=perl_structure
    &"['build_kanji.lr','eids.u2ff0._1.1_',[";
  begingroup
    save t;
    transform t[];
    yypart t1=yypart t2=1;
    ypart t1=yxpart t1=xypart t1=ypart t2=yxpart t2=xypart t2=0;
    (50,0) transformed t1=(50,0);
    (950,0) transformed t1=(splitpoint+overlap/2,0);
    (50,0) transformed t2=(splitpoint-overlap/2,0);
    (950,0) transformed t2=(950,0);
    tsu_xform(t1)(leftstuff);
    perl_structure:=perl_structure&"],[";
    tsu_xform(t2)(rightstuff);
  endgroup;
  perl_structure:=perl_structure&"]],";
enddef;

vardef build_kanji.road_enclose(text contents) =
  push_pbox_toexpand("build_kanji.road_enclose");
  perl_structure:=perl_structure&"'eids.u2FFA.u2ECC._3',";
  begingroup
    save myxf;
    transform myxf;
    (50,850) transformed myxf=(315,850);
    (50,-50) transformed myxf=(315,50);
    (950,-50) transformed myxf=(950,50);
    tsu_xform(myxf)(contents);
    push_stroke((100,770)..tension 1.2..(180,690)..(220,630),
      (1,1)--(1.3,1.3)--(1.9,1.9));
    set_bosize(0,92);
    push_stroke((80,453)--(240,450)..(mincho[230,210],250)
       ..tension 1.2..(mincho[180,140],50)..{curl 0.4}(60,-40),
      (1.4,1.4)--(1.6,1.6)--(1.4,1.4)--(1.2,1.2)--(1,1));
    set_botip(0,1,1);
    set_botip(0,2,1);
    set_boserif(0,1,4);
    set_bosize(0,92);
    push_stroke((point (2.3+mincho) of get_strokep(0))
         {(1-2*mincho)*direction (2.3+mincho) of get_strokep(0)}..
         (240,100)..(270,45+15*mincho)..(400,-10)..tension 3..(950,-20),
      (1,1)--(1.1,1.1)--(1.1,1.1)--(1.7,1.7)--(1.9,1.9));
    set_bosize(0,92);
  endgroup;
  expand_pbox;
enddef;

vardef build_kanji.sscale(text tran)(text curves) =
  tsu_xform(identity shifted (-centre_pt) tran shifted centre_pt)(curves);
enddef;

vardef build_kanji.spread_legs(expr dist)(text curves) =
  begingroup
    save osp;
    numeric osp;
    osp:=sp;
    curves;
    save mytr;
    transform mytr[];
    (50,-50) transformed mytr1=(50,-50);
    (500,-50) transformed mytr1=(500-dist/2,-50);
    (500,850) transformed mytr1=(500-dist/2,850);
    (950,-50) transformed mytr2=(950,-50);
    (500,-50) transformed mytr2=(500+dist/2,-50);
    (500,850) transformed mytr2=(500+dist/2,850);
    save i;
    i:=0;
    forever:
      exitif find_stroke(i)<osp;
      if xpart 0.75[llcorner get_strokep(i),urcorner get_strokep(i)]<475:
        replace_strokep(i,oldp transformed mytr1);
      elseif xpart 0.25[llcorner get_strokep(i),urcorner get_strokep(i)]>525:
        replace_strokep(i,oldp transformed mytr2);
      fi;
      i:=i-1;
    endfor;
  endgroup;
enddef;

vardef build_kanji.steam_enclose(expr ur)(text contents) =
  push_pbox_toexpand("build_kanji.steam_enclose");
  begingroup
    save xfa,xfb,xfc,xfd;
    transform xfa,xfb,xfc,xfd;
    (50,-50) transformed xfc=(50,-50);
    (950,850) transformed xfc=ur;
    xypart xfc=yxpart xfc=0;
    tsu_xform(xfc)(contents);

    (0,0) transformed xfa=(0,950) transformed xfc;
    (1,1) transformed xfa=(280,810);
    xypart xfa=yxpart xfa=0;
    (0,0) transformed xfb=(1100,950) transformed xfc;
    (1,1) transformed xfb=(970,810);
    xypart xfb=yxpart xfb=0;
    (0,1) transformed xfd=(1100,950) transformed xfc;
    (1,0) transformed xfd=(1000,-50);
    xypart xfd=yxpart xfd=0;

    push_stroke(((1,1)..tension 1.2..(0.5,0.45)..(0,0.2)) transformed xfa,
      (1.7,1.7)--(1.5,1.5)--(1.2,1.2));
    set_boserif(0,0,10);
    set_bosize(0,90);

    push_stroke((get_strokep(0) intersectionpoint
        (((0,0.8)--(1,0.8)) transformed xfa))--
        ((0.5,0.8) transformed xfb),
      (1.5,1.5)--(1.6,1.6));
    set_boserif(0,1,9);
    set_bosize(0,90);

    push_stroke(((0.8,0.4) transformed xfa)--((0.15,0.4) transformed xfb),
      (1.6,1.6)--(1.6,1.6));
    set_boserif(0,1,9);
    set_bosize(0,90);

    push_stroke(((0.2,0) transformed xfa)--(interpath(mincho,
      (0,1)..tension 1.6..(0.4,0)..(0.6,0)..tension 1.5..
        (0.73,0.2)..(0.8,0.4),
      (0,1)..tension 1.6..(0.25,0.2)..{right}(0.8,0){curl 1}..
       (0.6,0.2)..(0.6,0.4)
      ) transformed xfd),
      (1.6,1.6)–(1.6,1.6)–(1.4,1.4)–
        (1.4,1.4)–(1.2,1.2)–(0.9,0.9));
    set_boserif(0,1,4);
    set_botip(0,1,1);
    set_bosize(0,90);
  endgroup;
  expand_pbox;
enddef;

vardef build_kanji.tb(expr splitpoint,overlap)
  (text topstuff)(text bottomstuff) =
  perl_structure:=perl_structure
    &”[’build_kanji.tb”,eids.u2ff1._1.1_’,[”;
  begingroup
    save t;
    transform t[];
    xxpart t1=xxpart t2=1;
    xpart t1=xypart t1=yxpart t1=xpart t2=xypart t2=yxpart t2=0;
    (0,850) transformed t1=(0,850);
    (0,-50) transformed t1=(0,splitpoint-overlap/2);
    (0,850) transformed t2=(0,splitpoint+overlap/2);
    (0,-50) transformed t2=(0,-50);
    tsu_xform(t1)(topstuff);
    perl_structure:=perl_structure&”],[”;
    tsu_xform(t2)(bottomstuff);
  endgroup;
  perl_structure:=perl_structure&”]],”;
enddef;

vardef build_kanji.tcb(expr splitpointa,overlapa)(expr splitpointb,overlapb)
  (text topstuff)(text centrestuff)(text bottomstuff) =
  perl_structure:=perl_structure
    &”[’build_kanji.tcb”,eids.u2ff3._2.1_1.2_’,[”;
  begingroup
    save t;
    transform t[];
    xxpart t1=xxpart t2=xxpart t3=1;
    xpart t1=xypart t1=yxpart t1=0;
    xpart t2=xypart t2=yxpart t2=0;
    xpart t3=xypart t3=yxpart t3=0;
    (0,850) transformed t1=(0,850);
    (0,-50) transformed t1=(0,splitpointa-overlapa/2);
    (0,850) transformed t2=(0,splitpointa+overlapa/2);
    (0,-50) transformed t2=(0,splitpointb-overlapb/2);
    (0,850) transformed t3=(0,splitpointb+overlapb/2);
    (0,-50) transformed t3=(0,-50);
    tsu_xform(t1)(topstuff);
    perl_structure:=perl_structure&”],[”;
    tsu_xform(t2)(centrestuff);
    perl_structure:=perl_structure&”],[”;
    tsu_xform(t3)(bottomstuff);
  endgroup;
  perl_structure:=perl_structure&”]],”;
enddef;

vardef build_kanji.thickness(expr amount)(text curves) =
  begingroup
    save xsp;
    xsp:=sp;
    curves;
    i:=0;
    forever:
      exitif find_stroke(i)<xsp;
      if unknown get_bosize(i):
        set_bosize(i,100);
      fi;
      if get_bosize(i)>0:
        set_bosize(i,get_bosize(i)*amount);
      fi;
      i:=i-1;
    endfor;
  endgroup;
enddef;

vardef build_kanji.tricluster(expr topxscale)
  (text topstuff)(text leftstuff)(text rightstuff) =
  build_kanji.tb(500,0)
    (build_kanji.sscale(xscaled topxscale)(topstuff))
    (build_kanji.lr(480,0)
      (leftstuff)
      (rightstuff));
enddef;

vardef build_kanji.wind_enclose(text contents) =
  push_pbox_toexpand(”build_kanji.wind_enclose”);
  push_stroke((50,-50)..(120,100)..(160,300)..tension 1.2..(180,760)
      --interpath(mincho,
      (770,760){down}..{dir -72}(810,-20)..(860,-10)..tension 1.5..
        (890,100)..(910,200),
      (760,760){down}..(780,100)..{right}(910,-40){curl 1}..
        (890,60)..(890,160)),
    (1,1)--(1.3,1.3)--(1.5,1.5)--(1.6,1.6)--(1.6,1.6)--
      (1.4,1.4)--(1.4,1.4)--(1.2,1.2)--(0.9,0.9));
  set_botip(0,3,1);
  set_botip(0,4,1);
  set_boserif(0,3,4);
  set_boserif(0,4,4);
  begingroup
    save t;
    transform t;
    (50,-50) transformed t=(230,-50);
    (950,-50) transformed t=(700,-50);
    (950,850) transformed t=(700,660+20*mincho);
    tsu_xform(t)(contents);
  endgroup;
  expand_pbox;
enddef;
```

dakuten.mp

```
%
% Dakuten and handakuten for Tsukurimashou
% Copyright (C) 2011 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(dakuten);

--------------------------------------------------------------------------------

vardef dakuten(expr xf) =
  push_pbox_explicit(”dakuten”,
    identity shifted (-0.4,-0.5) scaled 200 rotated -50 transformed xf);

  push_stroke(((-80,10)..(-35,-35)..(10,-90)) transformed xf,
    (1,1)..(1.4,1.4)..(1.8,1.8));
  set_bosize(0,85);
  set_boserif(0,2,4);

  push_stroke(((0,80)..(50,30)..(100,-30)) transformed xf,
    (1,1)..(1.4,1.4)..(1.8,1.8));
  set_bosize(0,85);
  set_boserif(0,2,4);
enddef;

vardef handakuten(expr location) =
   push_lcblob(fullcircle scaled handakuten_outer shifted location
     transformed inverse tsu_rescale_xform);
   push_lcblob(reverse fullcircle scaled handakuten_inner shifted location
     transformed inverse tsu_rescale_xform);
enddef;
```

enclosed.mp

```
%
% Enclosed characters for Tsukurimashou
% Copyright (C) 2011, 2012 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(enclosed);

```

```

vardef circle.single =
  Fill fullcircle scaled (810+53*tsu_brush_max) shifted centre_pt;
  unFill reverse fullcircle scaled (810-53*tsu_brush_max) shifted centre_pt;
enddef;

vardef circle.double =
  Fill fullcircle scaled (880+60*tsu_brush_max) shifted centre_pt;
  unFill reverse fullcircle scaled (880-20*tsu_brush_max) shifted centre_pt;
  Fill fullcircle scaled (740+40*tsu_brush_max) shifted centre_pt;
  unFill reverse fullcircle scaled (740-40*tsu_brush_max) shifted centre_pt;
enddef;

vardef square.single(text curves) =
  tsu_xform(tsu_xf.sletter shifted tsu_xf.circ_slant_shift)(curves);
  push_stroke(
    ((500,790)--(100,790)--(100,-10)--(900,-10)--(900,790)--cycle)
      transformed inverse tsu_slant_xform,
    (2,2)--(2,2)--(2,2)--(2,2)--(2,2)--cycle);
  set_bosize(0,80);
  set_botip(0,1,1);
  set_botip(0,2,1);
  set_botip(0,3,1);
  set_botip(0,4,1);
enddef;

```

```

transform tsu_xf.circled;
xxpart tsu_xf.circled=yypart tsu_xf.circled=0.68;
xypart tsu_xf.circled=yxpart tsu_xf.circled=0;
centre_pt transformed tsu_xf.circled=centre_pt;

transform tsu_xf.cletter;
xxpart tsu_xf.cletter=yypart tsu_xf.cletter=0.56;
xypart tsu_xf.cletter=yxpart tsu_xf.cletter=0;
centre_pt transformed tsu_xf.cletter=centre_pt;
```

```

transform tsu_xf.ctwo.left;
xxpart tsu_xf.ctwo.left=yypart tsu_xf.ctwo.left=0.48;
xypart tsu_xf.ctwo.left=yxpart tsu_xf.ctwo.left=0;
(centre_pt+290*right) transformed tsu_xf.ctwo.left=centre_pt;

transform tsu_xf.ctwo.right;
xxpart tsu_xf.ctwo.right=yypart tsu_xf.ctwo.right=0.48;
xypart tsu_xf.ctwo.right=yxpart tsu_xf.ctwo.right=0;
(centre_pt+310*left) transformed tsu_xf.ctwo.right=centre_pt;

transform tsu_xf.sletter;
xxpart tsu_xf.sletter=yypart tsu_xf.sletter=0.71;
xypart tsu_xf.sletter=yxpart tsu_xf.sletter=0;
centre_pt transformed tsu_xf.sletter=centre_pt+10*up;

vardef tsu_xf.circ_slant_shift =
  (centre_pt-(centre_pt transformed tsu_slant_xform))
enddef;

────────────────────────────────────────────────────────────

transform tsu_xf.cbound;
tsu_xf.cbound=identity scaled 340 shifted centre_pt;
```

genjimon.mp

```
%
% Genjimon glyphs
% Copyright (C) 2011, 2012 Matthew Skala
%
[Standard copyright notice]

```

```

genji_grid:=150;

if unknown tsu_brush_max:
  if known brush_max:
    tsu_brush_max:=brush_max;
  else:
    tsu_brush_max:=0.75;
  fi;
fi;
if unknown genji_hw:
  genji_hw:=tsu_brush_max/1.5;
  if genji_hw>0.85: genji_hw:=0.85; fi;
fi;
if unknown genji_outline:
  boolean genji_outline;
  genji_outline:=false;
fi;
if genji_outline: genji_hw:=1-genji_hw; fi;
if unknown genji_rounded:
  boolean genji_rounded;
  genji_rounded:=false;
fi;

path genji_background;

% gb(f) - start a line at the bottom in file f
vardef gb(expr f,gp) =
  begingroup
    save myxf,mygl;
    transform myxf;
    path mygl;
    myxf=identity scaled (genji_grid/2) shifted (whatever,whatever);
    ((3-f)*2,4) transformed myxf=centre_pt;
    if genji_rounded:
      mygl:=((0,-genji_hw){right}..(genji_hw,0){up}..
          {up}(gp shifted (0,1)){down}..
          {down}(-genji_hw,0)..{right}cycle) transformed myxf;
    else:
```

```
  mygl:=((0,-genji_hw)--(genji_hw,-genji_hw)--
      (gp shifted (0,1))--
      (-genji_hw,-genji_hw)--cycle) transformed myxf;
fi;
if genji_outline:
  unFill reverse mygl;
  save mybk,x,y,old_dir;
  path mybk;
  pair old_dir;
  mybk:=(0,genji_hw-1.99) transformed myxf;
  old_dir:=right;
  for i:=1 upto (length mygl)-1:
    numeric x[],y[];
    z1=(point i of mygl)-(precontrol i of mygl);
    z2=(postcontrol i of mygl)-(point i of mygl);
    z3=z1/abs(z1);
    z4=z2/abs(z2);
    if z3=z4:
      mybk:=mybk{old_dir}..
        {z3}((0.99-genji_hw)*genji_grid*(z4 rotated -90)
            +point i of mygl);
    else:
      mybk:=mybk{old_dir}..
        {z3}((0.99-genji_hw)*genji_grid*((z3+z4) rotated -90)
            +point i of mygl);
    fi;
    if (((point i of mybk)-(point (i-1) of mybk)) dotprod z3<0)
    or (((point i of mybk)-(point (i-1) of mybk)) dotprod old_dir<0):
      mybk:=(subpath (0,i-1) of mybk)--(point i of mybk);
    fi;
    if (length mybk)>3:
      z5=(subpath ((length mybk)-4,(length mybk)-3) of mybk)
          intersectiontimes
        (subpath ((length mybk)-1,(length mybk)) of mybk);
      if x5>0:
        mybk:=(subpath (0,(length mybk)-4+x5) of mybk)..
          (subpath ((length mybk)-1+y5,infinity) of mybk)
      fi;
    fi;
    if (length mybk)>3:
      z6=(subpath ((length mybk)-4,(length mybk)-3) of mybk)
          intersectiontimes
        (subpath ((length mybk)-2,(length mybk)-1) of mybk);
      if x6>0:
        mybk:=(subpath (0,(length mybk)-4+x6) of mybk)..
          (subpath ((length mybk)-2+y6,infinity) of mybk)
      fi;
    fi;
```

```
        if (length mybk)>3:
          z7=((point (length mybk)-4 of mybk)
            -(precontrol (length mybk)-4 of mybk));
          z8=((postcontrol (length mybk)-3 of mybk)
            -(point (length mybk)-3 of mybk));
          if (abs(z7)>0) and (abs(z8)>0):
            if (z7/abs(z7)) dotprod (z8/abs(z8))<-0.1:
              mybk:=(subpath (0,(length mybk)-4) of mybk)--
                (subpath ((length mybk)-3,infinity) of mybk);
            fi;
          fi;
        fi;
        old_dir:=z4;
      endfor;
      mybk:=regenerate(mybk{old_dir}..{right}cycle);
      dangerousFill mybk;
    else:
      Fill mygl;
    fi;
  endgroup;
enddef;

path ge_path[];
ge_path[0]=(genji_hw,0)--(genji_hw,genji_hw)--
  (-genji_hw,genji_hw)--(-genji_hw,0);
ge_path[1]=(genji_hw,0){up}..(0,genji_hw){left}..
  (-genji_hw,genji_hw)--(-genji_hw,0);
ge_path[2]=(genji_hw,0)--(genji_hw,genji_hw)--
  (0,genji_hw){left}..{down}(-genji_hw,0);
ge_path[3]=(genji_hw,0){up}..(0,genji_hw){left}..{down}(-genji_hw,0);

% ge(t) - end a line, style t
vardef ge(expr t) =
  if genji_rounded:
    ((genji_hw,0)..(ge_path[t] shifted (0,1))..(-genji_hw,0))
  else:
    ((genji_hw,0)..(ge_path[0] shifted (0,1))..(-genji_hw,0))
  fi
enddef;

% gf(d) - go forward d steps
vardef gf(expr d,gp) =
  ((genji_hw,0)--(gp shifted (0,2*d))--(-genji_hw,0))
enddef;

% gr(r) - turn to right, radius r
vardef gr(expr r,gp) =
  if genji_rounded and (r>=0):
    ((genji_hw,0){up}..
     (gp shifted (0,r+1) rotated -90 shifted (0,r+1))..
     {down}(-genji_hw,0))
  else:
    ((genji_hw,0)--(genji_hw,max(r,0)+1-genji_hw)--
     (gp shifted (0,max(r,0)+1) rotated -90 shifted (0,max(r,0)+1))--
     (-genji_hw,max(r,0)+1+genji_hw)--(-genji_hw,0))
  fi
enddef;

% gl(r) - turn to left, radius r
vardef gl(expr r,gp) =
  if genji_rounded and (r>=0):
    ((genji_hw,0)..
     (gp shifted (0,r+1) rotated 90 shifted (0,r+1))..
     (-genji_hw,0))
  else:
    ((genji_hw,0)--(genji_hw,max(r,0)+1+genji_hw)--
     (gp shifted (0,max(r,0)+1) rotated 90 shifted (0,max(r,0)+1))--
     (-genji_hw,max(r,0)+1-genji_hw)--(-genji_hw,0))
  fi
enddef;

% gt(gpa,gpb) - make a T-junction
vardef gt(expr gpa,gpb) =
  ((genji_hw,0)--(genji_hw,1-genji_hw)--
   (gpa shifted (0,1) rotated -90 shifted (0,1))--
   (genji_hw,1+genji_hw)--(gpb shifted (0,2))--(-genji_hw,0))
enddef;

% gx(gpa,gpb,gpc) - make an X-junction
vardef gx(expr gpa,gpb,gpc) =
  ((genji_hw,0)--(genji_hw,1-genji_hw)--
   (gpa shifted (0,1) rotated -90 shifted (0,1))--
   (genji_hw,1+genji_hw)--(gpb shifted (0,2))--(-genji_hw,1+genji_hw)--
   (gpc shifted (0,1) rotated 90 shifted (0,1))--
   (-genji_hw,1-genji_hw)--(-genji_hw,0))
enddef;



% #1 Kiritsubo
vardef genjimon.kiritsubo =
  gb(1,gf(2.5,gr(1,gr(1,gf(2.5,ge(3))))));
  gb(2,gf(2,ge(3)));
  gb(4,gf(3,gr(0,gr(0,gf(3,ge(3))))));
enddef;

```

```
% #2 Hahakigi
vardef genjimon.hahakigi =
  gb(1,gf(3,ge(3)));
  gb(2,gf(3,ge(3)));
  gb(3,gf(3,ge(3)));
  gb(4,gf(3,ge(3)));
  gb(5,gf(3,ge(3)));
enddef;

% #3 Utsusemi
vardef genjimon.utsusemi =
  gb(1,gf(3,ge(3)));
  gb(2,gf(3,ge(3)));
  gb(3,gf(3,ge(3)));
  gb(4,gf(3,gr(0,gr(0,gf(3,ge(3))))));
enddef;

% #4 Yuugao
vardef genjimon.yuugao =
  gb(1,gf(3,ge(3)));
  gb(2,gf(3,ge(3)));
  gb(3,gf(3,gr(0,gr(0,gf(3,ge(3))))));
  gb(5,gf(3,ge(3)));
enddef;

% #5 Wakamurasaki
vardef genjimon.wakamurasaki =
  gb(1,gf(3,ge(3)));
  gb(2,gf(3,gr(0,gr(0,gf(3,ge(3))))));
  gb(4,gf(3,gr(0,gr(0,gf(3,ge(3))))));
enddef;

% #6 Suetsumuhana
vardef genjimon.suetsumuhana =
  gb(1,gf(3,gr(0,gt(gf(3,ge(3)),gt(gf(3,ge(3)),
    gr(0,gf(3,ge(3))))))));
  gb(5,gf(3,ge(3)));
enddef;

% #7 Momiji no Ga
vardef genjimon.momiji_no_ga =
  gb(1,gf(3,ge(3)));
  gb(4,gf(2,ge(1)));
  gb(2,gf(3,gr(0,gt(gf(3,ge(3)),gf(0.5,gr(1,gf(2.5,ge(3))))))));
enddef;

% #8 Hana no En
vardef genjimon.hana_no_en =
  gb(1,gf(3,ge(3)));
  gb(2,gf(3,ge(3)));
  gb(3,gf(2.5,gr(1,gr(1,gf(2.5,ge(3))))));
  gb(4,gf(2,ge(3)));
enddef;

% #9 Aoi
vardef genjimon.aoi =
  gb(1,gf(3,gr(0,gr(0,gf(3,ge(3))))));
  gb(3,gf(3,ge(3)));
  gb(4,gf(3,ge(3)));
  gb(5,gf(3,ge(3)));
enddef;

% #10 Sakaki
vardef genjimon.sakaki =
  gb(1,gf(3,gr(0,gt(gf(3,ge(3)),gr(0,gf(3,ge(3)))))));
  gb(4,gf(3,gr(0,gr(0,gf(3,ge(3))))));
enddef;

% #11 Hana Chiru Sato
vardef genjimon.hana_chiru_sato =
  gb(1,gf(3,ge(3)));
  gb(2,gf(2,gr(2,gr(0,gx(gl(0,gf(2,ge(3))),
    gf(2,ge(3)),gr(0,gf(2,ge(3))))))));
enddef;

% #12 Suma
vardef genjimon.suma =
  gb(1,gf(2,gr(0,gx(gf(2,ge(3)),gt(gf(2,ge(3)),
    gr(0,gf(2,ge(3)))),gr(0,gf(1,gr(2,gf(2,ge(3)))))))));
enddef;

% #13 Akashi
vardef genjimon.akashi =
  gb(1,gf(3,ge(3)));
  gb(2,gf(3,gr(0,gr(0,gf(3,ge(3))))));
  gb(4,gf(3,ge(3)));
  gb(5,gf(3,ge(3)));
enddef;

% #14 Miotsukushi
vardef genjimon.miotsukushi =
  gb(1,gf(3,ge(3)));
  gb(2,gf(2.5,gr(1,gf(0.5,gt(gf(3,ge(3)),gr(0,gf(3,ge(3))))))));
  gb(3,gf(2,ge(2)));
enddef;

```

```
% #15 Yomogyuu
vardef genjimon.yomogyuu =
  gb(1,gf(3,gr(0,gt(gf(3,ge(3)),gr(0,gf(3,ge(3)))))));
  gb(4,gf(3,ge(3)));
  gb(5,gf(3,ge(3)));
enddef;

% #16 Sekiya
vardef genjimon.sekiya =
  gb(1,gf(3,ge(3)));
  gb(2,gf(3,gr(0,gt(gf(3,ge(3)),gr(0,gf(3,ge(3)))))));
  gb(5,gf(3,ge(3)));
enddef;

% #17 Eawase
vardef genjimon.eawase =
  gb(1,gf(2,gr(0,gx(gf(2,ge(3)),gr(-1,gf(2,ge(3))),
    gr(0,gf(1.5,gr(1,gf(2.5,ge(3))))))))));
  gb(4,gf(2,ge(1)));
enddef;

% #18 Matsukaze
vardef genjimon.matsukaze =
  gb(1,gf(3,gr(0,gr(0,gf(3,ge(3))))));
  gb(3,gf(3,gr(0,gr(0,gf(3,ge(3))))));
  gb(5,gf(3,ge(3)));
enddef;

% #19 Usugumo
vardef genjimon.usugumo =
  gb(1,gf(3,ge(3)));
  gb(2,gf(3,gr(0,gt(gf(3,ge(3)),gt(gf(3,ge(3)),
    gr(0,gf(3,ge(3)))))))));
enddef;

% #20 Asagao
vardef genjimon.asagao =
  gb(1,gf(2.5,gr(1,gf(0.5,gt(gf(3,ge(3)),gr(0,gf(3,ge(3))))))));
  gb(2,gf(2,ge(2)));
  gb(5,gf(3,ge(3)));
enddef;

% #21 Otome
vardef genjimon.otome =
  gb(1,gf(2.5,gr(1,gr(1,gf(2.5,ge(3))))));
  gb(2,gf(2,ge(3)));
  gb(4,gf(3,ge(3)));
  gb(5,gf(3,ge(3)));
```

```
enddef;

% #22 Tamakazura
vardef genjimon.tamakazura =
  gb(1,gf(3,gr(0,gr(0,gf(3,ge(3))))));
  gb(3,gf(3,gr(0,gt(gf(3,ge(3)),gr(0,gf(3,ge(3)))))));
enddef;

% #23 Hatsune
vardef genjimon.hatsune =
  gb(1,gf(2,gr(0,gx(gf(2,ge(3)),gr(0,gf(2,ge(3))),
    gr(0,gr(2,gf(2,ge(3))))))));
  gb(5,gf(3,ge(3)));
enddef;

% #24 Kochou
vardef genjimon.kochou =
  gb(1,gf(2,gr(2,gf(1,gr(0,gx(reverse gt(gf(2,ge(3)),
    gr(0,gf(2,ge(3)))) xscaled -1,gf(2,ge(3)),
    gr(0,gf(2,ge(3)))))))));
enddef;

% #25 Hotaru
vardef genjimon.hotaru =
  gb(1,gf(3,gr(0,gt(gf(3,ge(3)),gf(0.5,gr(1,gf(2.5,ge(3))))))));
  gb(3,gf(2,ge(1)));
  gb(5,gf(3,ge(3)));
enddef;

% #26 Tokonatsu
vardef genjimon.tokonatsu =
  gb(1,gf(3,ge(3)));
  gb(2,gf(3,ge(3)));
  gb(3,gf(3,gr(0,gt(gf(3,ge(3)),gr(0,gf(3,ge(3)))))));
enddef;

% #27 Kagaribi
vardef genjimon.kagaribi =
  gb(1,gf(3,ge(3)));
  gb(2,gf(2.5,gr(1,gr(1,gf(2.5,ge(3))))));
  gb(3,gf(2,ge(3)));
  gb(5,gf(3,ge(3)));
enddef;

% #28 Nowaki
vardef genjimon.nowaki =
  gb(1,gf(3,gr(0,gr(0,gf(3,ge(3))))));
  gb(3,gf(3,ge(3)));
  gb(4,gf(3,gr(0,gr(0,gf(3,ge(3))))));
enddef;

% #29 Miyuki
vardef genjimon.miyuki =
  gb(1,gf(2,gr(2,gr(0,gx(gl(0,gf(2,ge(3))),
    gf(2,ge(3)),gt(gf(2,ge(3)),gr(0,gf(2,ge(3)))))))));
enddef;

% #30 Fujibakama
vardef genjimon.fujibakama =
  gb(1,gf(2.5,gr(1,gf(1,gr(1,gf(2.5,ge(3)))))));
  gb(2,gf(2,ge(2)));
  gb(3,gf(2,ge(1)));
  gb(5,gf(3,ge(3)));
enddef;

% #31 Makibashira
vardef genjimon.makibashira =
  gb(1,gf(1.5,gr(3,gr(3,gf(1.5,ge(3))))));
  gb(2,gf(1.5,gr(1,gr(1,gf(1.5,ge(3))))));
  gb(3,gf(1,ge(3)));
enddef;

% #32 Umegae
vardef genjimon.umegae =
  gb(1,gf(3,gr(0,gt(gf(3,ge(3)),gt(gf(3,ge(3)),gf(0.5,
    gr(1,gf(2.5,ge(3)))))))));
  gb(4,gf(2,ge(1)));
enddef;

% #33 Fuji no Uraba
vardef genjimon.fuji_no_uraba =
  gb(1,gf(3,ge(3)));
  gb(2,gf(2,gr(2,gr(2,gf(2,ge(3))))));
  gb(3,gf(2,gr(0,gr(0,gf(2,ge(3))))));
enddef;

% #34 Wakana no Jou
vardef genjimon.wakana_no_jou =
  gb(1,gf(3,gr(0,gt(gf(3,ge(3)),gf(1,
    gr(2,gf(2,ge(3))))))));
  gb(3,gf(2,gr(-1,gr(0,gf(2,ge(3))))));
enddef;

% #35 Wakana no Ge
vardef genjimon.wakana_no_ge =
  gb(1,gf(3,gr(0,gt(gf(3,ge(3)),gf(1,
    gr(0,gx(gl(-1,gf(2,ge(3))),gf(2,ge(3)),gr(0,gf(2,ge(3))))))))));
enddef;

% #36 Kashiwagi
vardef genjimon.kashiwagi =
  gb(1,gf(2.5,gr(1,gf(0.5,gt(gf(3,ge(3)),
    gf(0.5,gr(1,gf(2.5,ge(3)))))))));
  gb(2,gf(2,ge(2)));
  gb(4,gf(2,ge(1)));
enddef;

% #37 Yokobue
vardef genjimon.yokobue =
  gb(1,gf(2.5,gr(1,gf(1.5,gt(gf(3,ge(3)),
    gr(0,gf(3,ge(3))))))));
  gb(2,gf(2,ge(2)));
  gb(3,gf(2,ge(0)));
enddef;

% #38 Suzumushi
vardef genjimon.suzumushi =
  gb(1,gf(2.5,gr(1,gf(1.5,gr(2,gf(2,ge(3)))))));
  gb(2,gf(2,ge(2)));
  gb(3,gf(2,gr(-1,gr(0,gf(2,ge(3))))));
enddef;

% #39 Yuugiri
vardef genjimon.yuugiri =
  gb(1,gf(1.5,gr(1,gf(0.5,gx(gf(2,ge(3)),gr(0,gf(2,ge(3))),
    gr(0,gr(2,gf(2,ge(3)))))))));
  gb(2,gf(1,ge(2)));
enddef;

% #40 Minori
vardef genjimon.minori =
  gb(1,gf(1.5,gr(3,gf(0.5,gr(0,gx(gf(0.5,gl(1,gf(1.5,ge(3)))),
    gf(2,ge(3)),gr(0,gf(2,ge(3)))))))));
  gb(3,gf(1,ge(2)));
enddef;

% #41 Maboroshi
vardef genjimon.maboroshi =
  gb(1,gf(2.5,gr(1,gf(2,gr(1,gf(2.5,ge(3)))))));
  gb(2,gf(2,ge(2)));
  gb(3,gf(2,ge(0)));
  gb(4,gf(2,ge(1)));
enddef;

```

```
% #42 Ninounomiya
vardef genjimon.ninounomiya =
  gb(1,gf(2,gr(0,gt(gf(2,ge(3)),gx(gf(2,ge(3)),gr(0,gf(2,ge(3))),
    gr(0,gr(2,gf(2,ge(3)))))))))));
enddef;

% #43 Koubai
vardef genjimon.koubai =
  gb(1,gf(3,ge(3)));
  gb(2,gf(2.5,gr(1,gf(1,gr(1,gf(2.5,ge(3))))))));
  gb(3,gf(2,ge(2)));
  gb(4,gf(2,ge(1)));
enddef;

% #44 Takegawa
vardef genjimon.takegawa =
  gb(1,gf(2,gr(2,gf(1,gr(2,gf(2,ge(3))))))));
  gb(2,gf(2,gr(0,gt(gf(2,ge(3)),gr(0,gf(2,ge(3))))))));
enddef;

% #45 Hashihime
vardef genjimon.hashihime =
  gb(1,gf(2.5,gr(1,gf(0.5,gt(gf(3,ge(3)),gt(gf(3,ge(3)),
    gr(0,gf(3,ge(3))))))))));
  gb(2,gf(2,ge(2)));
enddef;

% #46 Shii ga Moto
vardef genjimon.shii_ga_moto =
  gb(1,gf(2,gr(2,gr(2,gf(2,ge(3)))))));
  gb(2,gf(2,gr(0,gr(0,gf(2,ge(3)))))));
  gb(5,gf(3,ge(3)));
enddef;

% #47 Agemaki
vardef genjimon.agemaki =
  gb(1,gf(2,gr(2,gf(1,gt(gf(3,ge(3)),gr(0,gf(3,ge(3)))))))));
  gb(2,gf(2,gr(0,gr(-1,gf(2,ge(3)))))));
enddef;

% #48 Sawarabi
vardef genjimon.sawarabi =
  gb(1,gf(3,gr(0,gr(0,gf(3,ge(3)))))));
  gb(3,gf(2.5,gr(1,gr(1,gf(2.5,ge(3)))))));
  gb(4,gf(2,ge(3)));
enddef;

% #49 Yadorigi
```

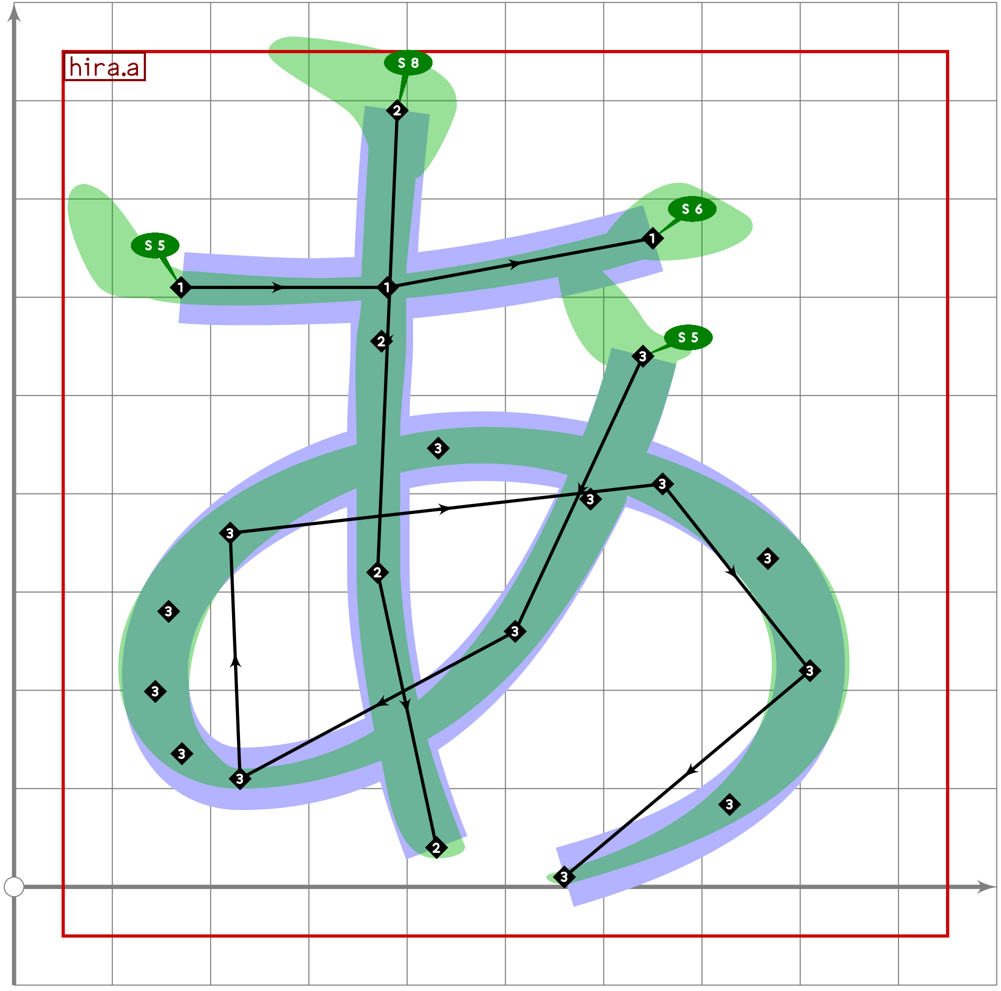
```
vardef  genjimon.yadorigi =
  gb(1,gf(3,gr(0,gt(gf(3,ge(3)),gf(1,gt(gf(3,ge(3)),
    gr(0,gf(3,ge(3))))))))));
  gb(3,gf(2,ge(0)));
enddef;

% #50  Azumaya
vardef  genjimon.azumaya =
  gb(1,gf(3,gr(0,gt(gf(3,ge(3)),gf(1.5,gr(1,gf(2.5,ge(3))))))));
  gb(3,gf(2,ge(0)));
  gb(4,gf(2,ge(1)));
enddef;

% #51 Ukifune
vardef  genjimon.ukifune =
  gb(1,gf(2,gr(2,gf(1,gf(0.5,gr(1,gf(2.5,ge(3))))))));
  gb(2,gf(2,gr(0,gr(-1,gf(2,ge(3))))));
  gb(4,gf(2,ge(1)));
enddef;

% #52  Kagerou
vardef  genjimon.kagerou =
  gb(1,gf(2,gr(2,gt(gx(gl(0,gf(2,ge(3))),
    gf(2,ge(3)),gr(0,gf(2,ge(3)))),gr(2,gf(2,ge(3))))))));
enddef;

% #53  Tenarai
vardef  genjimon.tenarai =
  gb(1,gf(3,gr(0,gt(gf(3,ge(3)),gt(gf(3,ge(3)),gt(gf(3,ge(3)),
    gr(0,gf(3,ge(3))))))))));
enddef;

% #54  Yume no Ukihashi
vardef  genjimon.yume__no__ukihashi =
  gb(1,gf(3,gr(0,gr(0,gf(3,gl(0,gl(0,gf(3,gr(0,gr(0,gf(3,
    gl(0,gl(0,gf(3,ge(3)))))))))))))));
enddef;
```

hiragana.mp

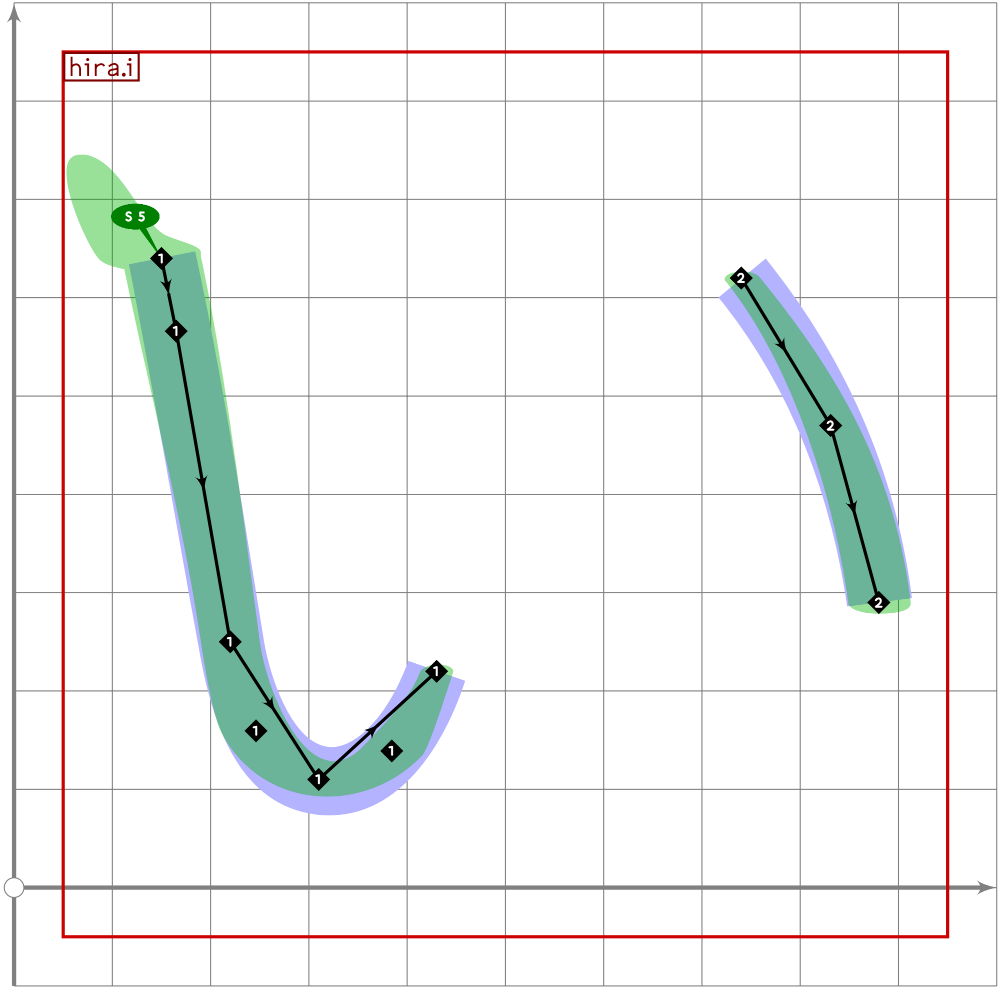
```
%
% Hiragana for Tsukurimashou
% Copyright (C) 2011, 2012 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(hiragana);

```

Hiragana Vowels

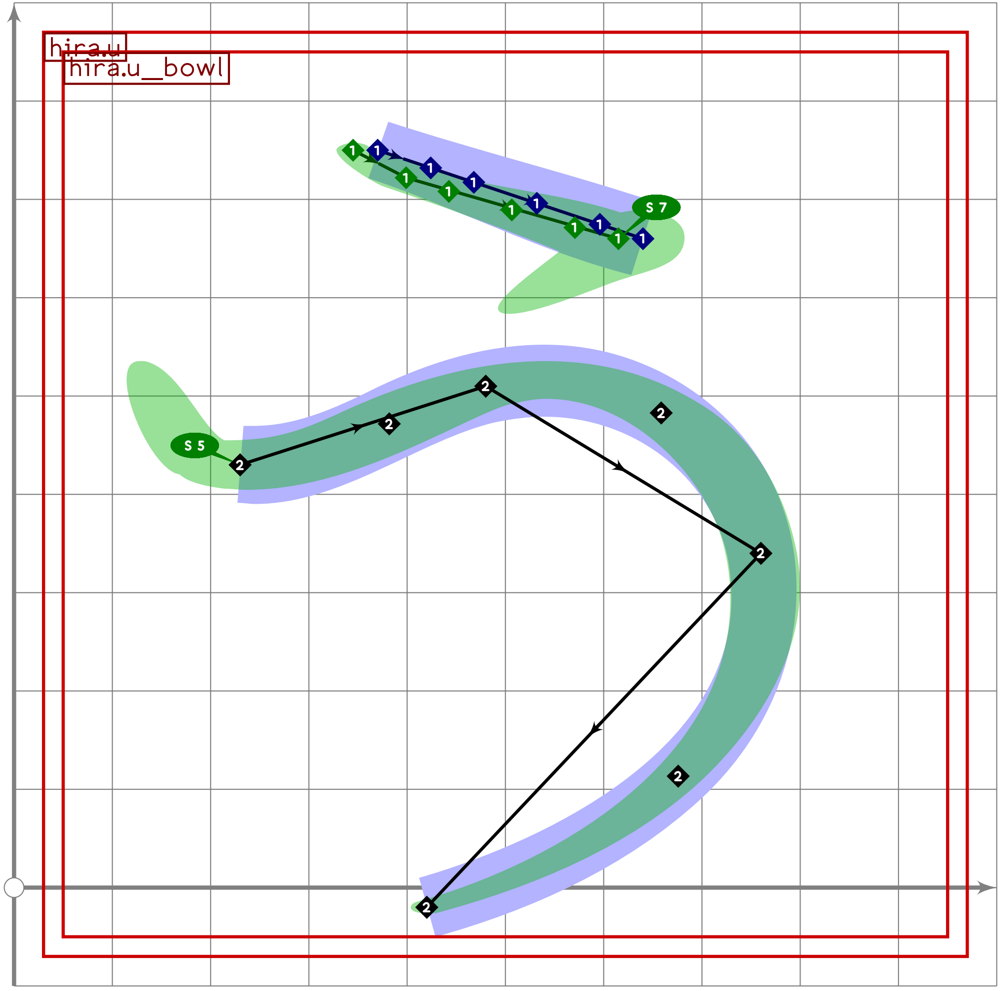
```
%%%%%%%%%% HIRAGANA VOWELS
```



```

vardef hira.a =
  push_pbox_toexpand("hira.a");

  push_stroke((170,610)..(380,610)..(650,660),
    (1.6,1.7)--(1.4,1.4)--(1.6,1.6));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  push_stroke((390,790)..tension 1.5..(370,320)..(430,40),
    (1.2,1.2)−(1.1,1.1)−(1.3,1.3));
  set_boserif(0,0,8);

  push_stroke((640,540)..tension 1.2..(510,260)..(230,110){left}..
      (220,360)..(660,410)..(810,220)..{curl 0}(560,10),
    (1.5,1.5)−(1.4,1.4)−(1.2,1.2)−
      (1.7,1.7)−(1.8,1.8)−(1.6,1.6)−(1,1));
  set_boserif(0,0,5);
  expand_pbox;
enddef;
```



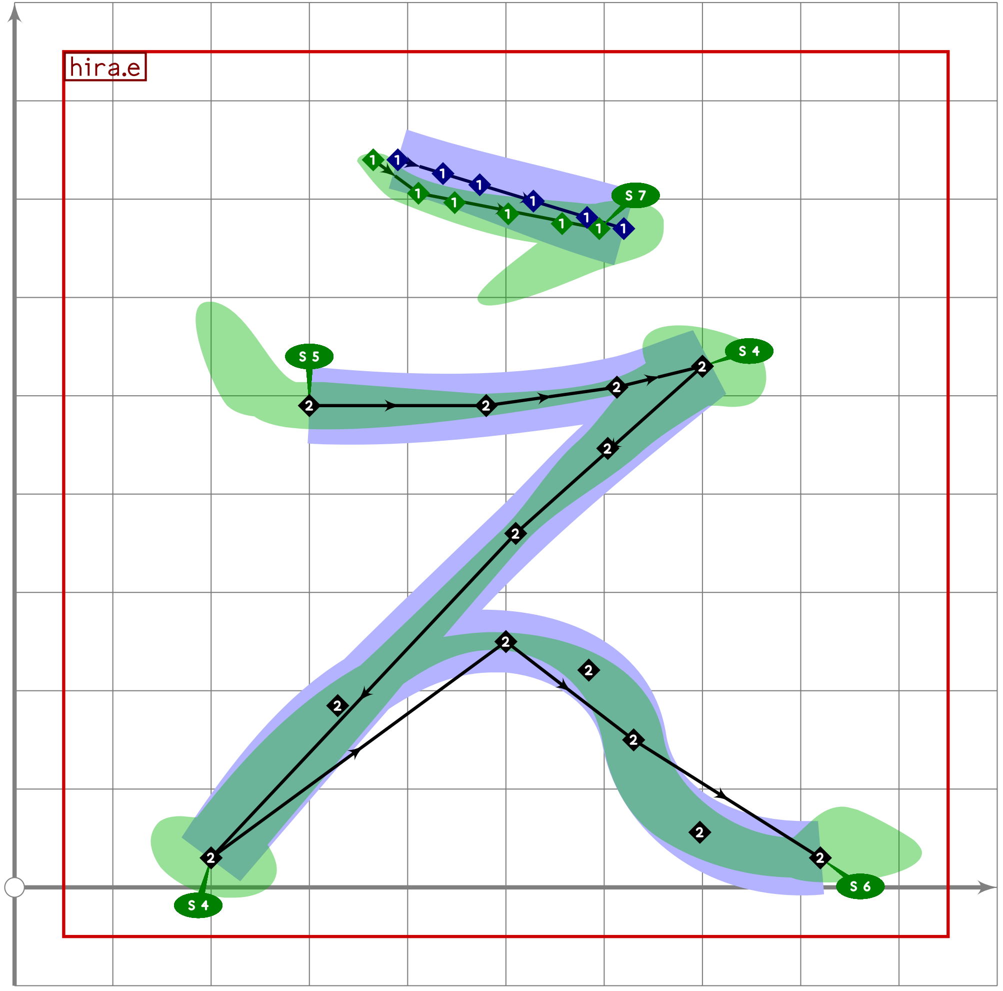
```

vardef hira.i =
  push_pbox_toexpand("hira.i");

  push_stroke((150,640)..(165,566)..(220,250)..(310,110)..{curl 0.1}(430,220),
    (1.6,1.6)..(1.6,1.6)..(1.3,1.3)..(1.8,1.8)..(1,1));
  set_boserif(0,0,5);

  push_stroke((740,620)..(831,470)..(880,290),
    (1,1)..(1.3,1.3)..(1.4,1.4));
  expand_pbox;
enddef;

vardef hira.u_bowl =
  push_pbox_toexpand("hira.u_bowl");

  push_stroke((230,430){dir 355}..(480,510)..(760,340)..{curl 0.1}(420,-20),
    (2.3,2.3)..(2,2)..(1.5,1.5)..(1,1));
  set_boserif(0,0,5);
  expand_pbox;
enddef;
```



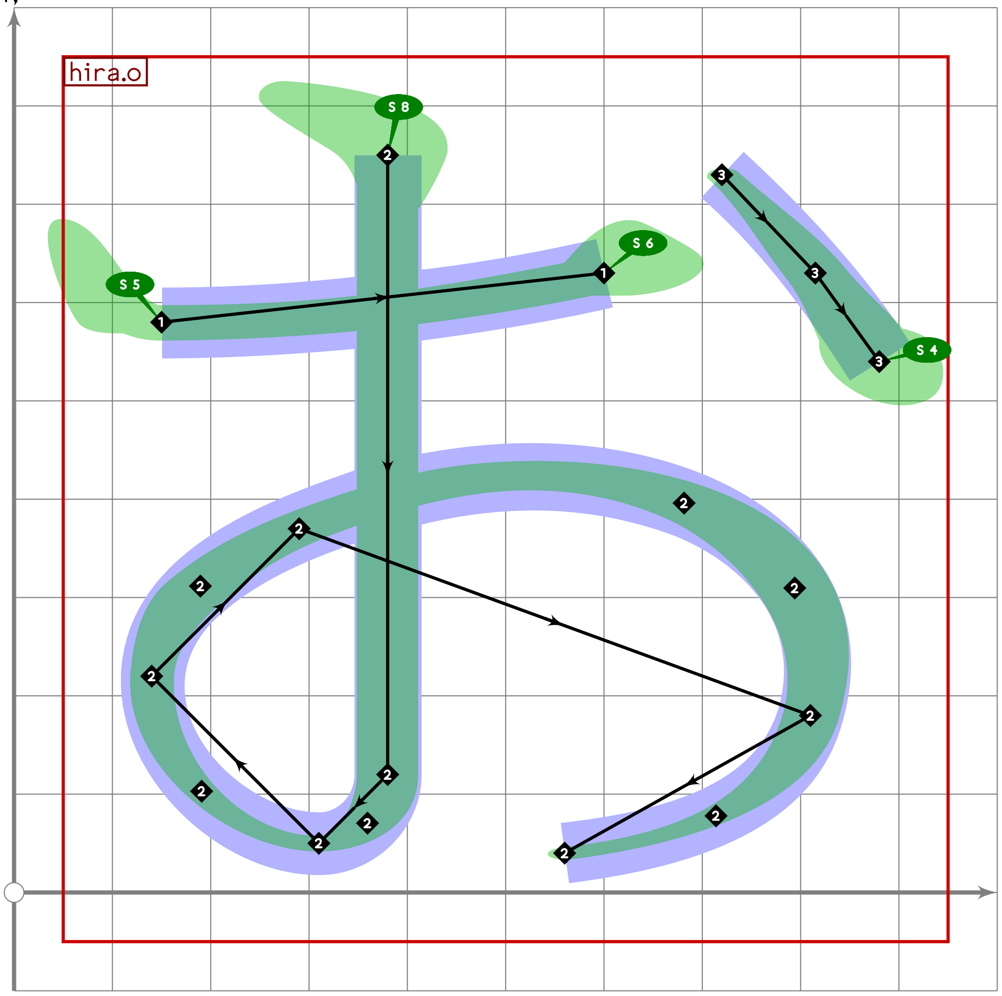
```

vardef hira.u =
  push_pbox_toexpand("hira.u");

  push_stroke(((370,750)..(0.2[(370,750),(640,660)]+10*down*mincho)..
    tension 2..(640,660)) shifted (25*left*mincho),
    (1,1)..(1.4,1.4)..(2.3,2.3));
```

U+3048
tsuku.uni3048

```
  set_boserif(0,2,7);

  hira.u_bowl;
  expand_pbox;
enddef;
```



```

vardef hira.e =
  push_pbox_toexpand("hira.e");

  push_stroke(((390,740)..(0.2[(390,740),(620,670)]+20*down*mincho)..
    tension 2..(620,670)) shifted (25*left*mincho),
    (1,1)..(1.4,1.4)..(2.3,2.3));
  set_boserif(0,2,7);

  push_stroke((300,490)..(480,490)..{curl 1}(700,530){curl 1}..
      (510,360)..{curl 1}(200,30){curl 0}..
      (500,250)..(630,150){dir 280}..{dir 5}(820,30),
    (2.2,2.2)--(1.3,1.3)--(1,1)--(2.01,2.01)
      --(1,1)--(1.7,1.7)
      --(1.2,1.2)--(1.4,1.4)--(2,2));
  replace_strokep(0)(insert_nodes(oldp)(1.6));
  set_botip(0,3,0);
  set_botip(0,5,0);
  set_boserif(0,0,5);
  set_boserif(0,3,4);
  set_boserif(0,5,4);
  set_boserif(0,8,6);
  expand_pbox;
enddef;
```



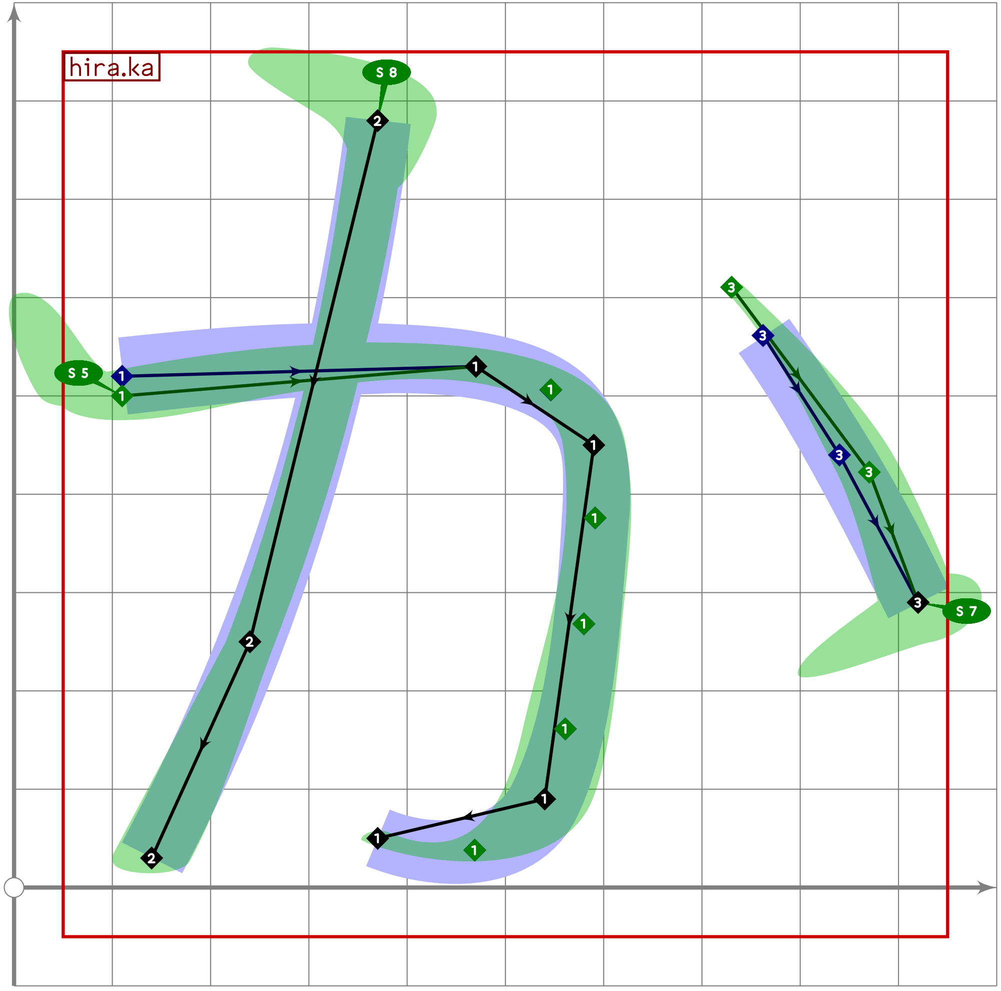
```

vardef hira.o =
  push_pbox_toexpand("hira.o");

  push_stroke((150,580){right}..(600,630),
    (1.8,1.8)..(1.8,1.8));
  set_boserif(0,0,5);
  set_boserif(0,1,6);

  push_stroke((380,750)..(380,120){down}..(310,50){left}..tension 1.1..
      (140,220)..(290,370)..(810,180)..{curl 0}(560,40),
    (1.4,1.4)..(1.3,1.3)..(1,1)..(1.5,1.5)..(1.6,1.6)..
      (1.6,1.6)..(1,1));
```

U+304B
tsuku.uni304B

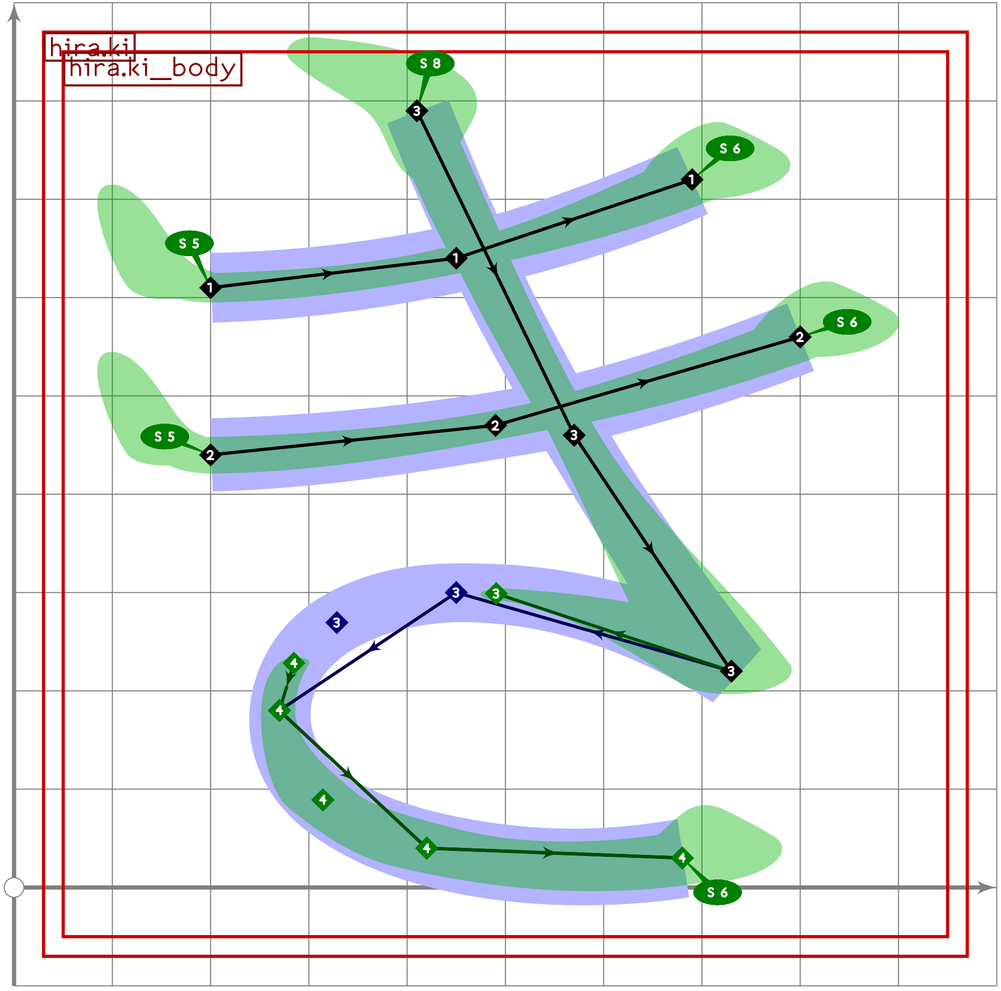
```
  set_boserif(0,0,8);

  push_stroke((720,730)..(815,630)..(880,540),
    (1,1)..(1.4,1.4)..(1.8,1.8));
  set_boserif(0,2,4);
  expand_pbox;
enddef;

```

Hiragana Kakikukeko/Gagigugego

```
%%%%%%%%%% HIRAGANA KAKIKUKEKO/GAGIGUGEGO
```

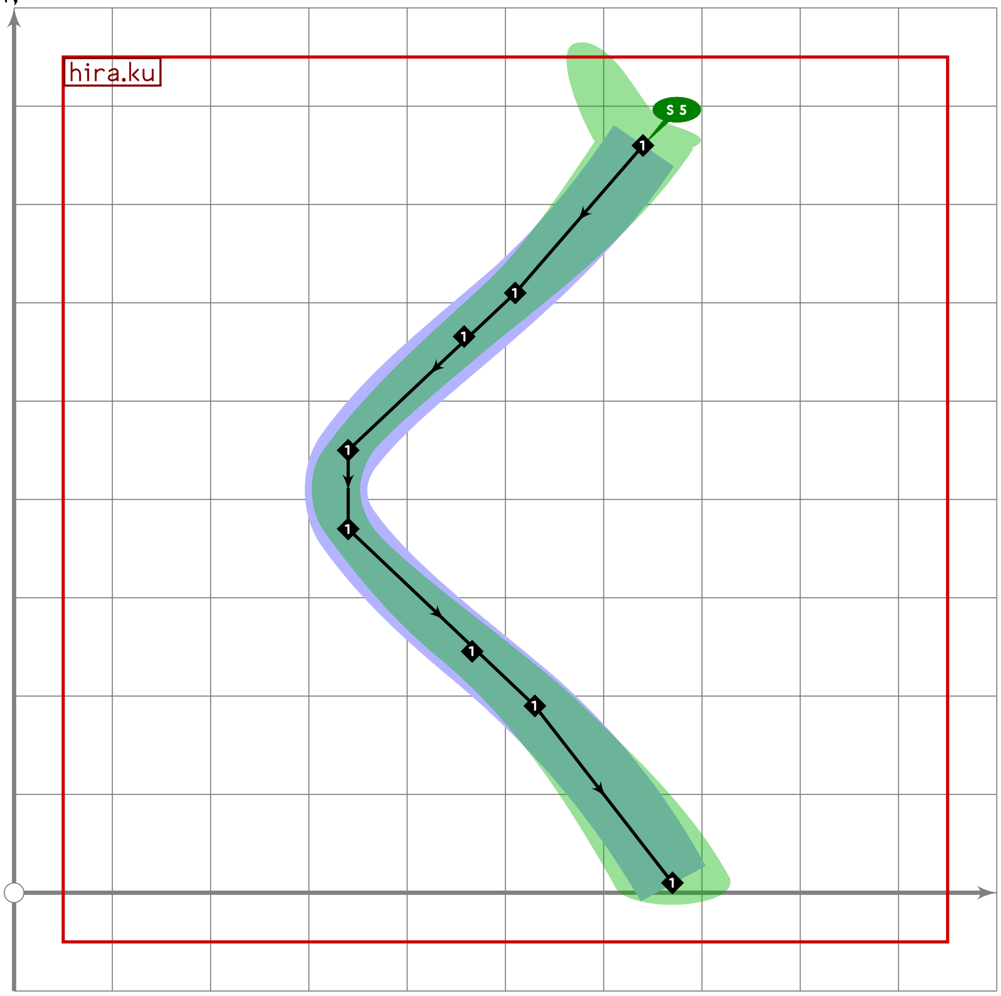


```

vardef hira.ka =
  push_pbox_toexpand("hira.ka");

  push_stroke(((110,520)+20*mincho*down){curl 0}..(470,530)..(590,450)..
      tension 2..(540,90)..{curl 0.3}(370,50),
    (2.3,2.3)..(1.7,1.7)..(1.4,1.4)..(1.8,1.8)..(1,1));
  set_boserif(0,0,5);

  push_stroke((370,780)..(240,250)..(140,30),
    (1.3,1.3)..(1.2,1.2)..(1.6,1.6));
  set_boserif(0,0,8);

  push_stroke((720,620)..((840,440)+35*mincho*(dir -30))..(920,290),
    (0.8,0.8)..(1.4,1.4)..(1.6,1.6));
  set_boserif(0,2,7);
  expand_pbox;
enddef;

vardef hira.ki_body =
  push_pbox_toexpand("hira.ki_body");

  push_stroke((410,790)..(570,460)..{curl 1}(730,220){curl 1}..
      (450,300)..(270,180)..(420,40)..(680,30),
    (1.4,1.4)--(1.2,1.2)--(2.3,2)--(0.60,1)..(0.9,1.1)..
      (2.1,2.1)..(2.4,2.4));
  set_botip(0,2,0);
  set_boserif(0,0,8);
  set_boserif(0,6,6);
  expand_pbox;
enddef;
```

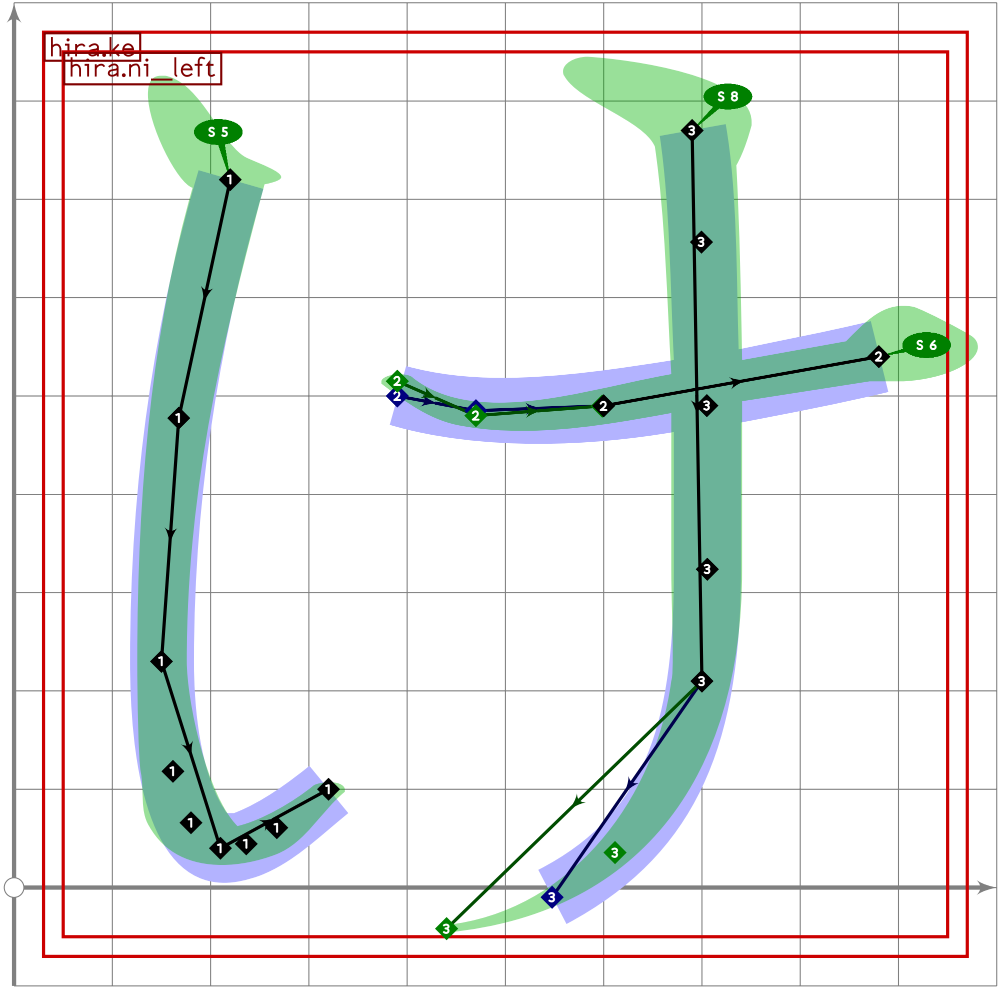


U+304F
tsuku.uni304F

```

vardef hira.ki =
  push_pbox_toexpand("hira.ki");

  push_stroke((200,610)..(450,640)..(690,720),
    (1.6,1.6)--(1.4,1.4)--(1.9,1.9));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  push_stroke((200,440)..(490,470)..(800,560),
    (1.9,1.9)--(1.5,1.5)--(1.9,1.9));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  hira.ki_body;
  expand_pbox;
enddef;
```

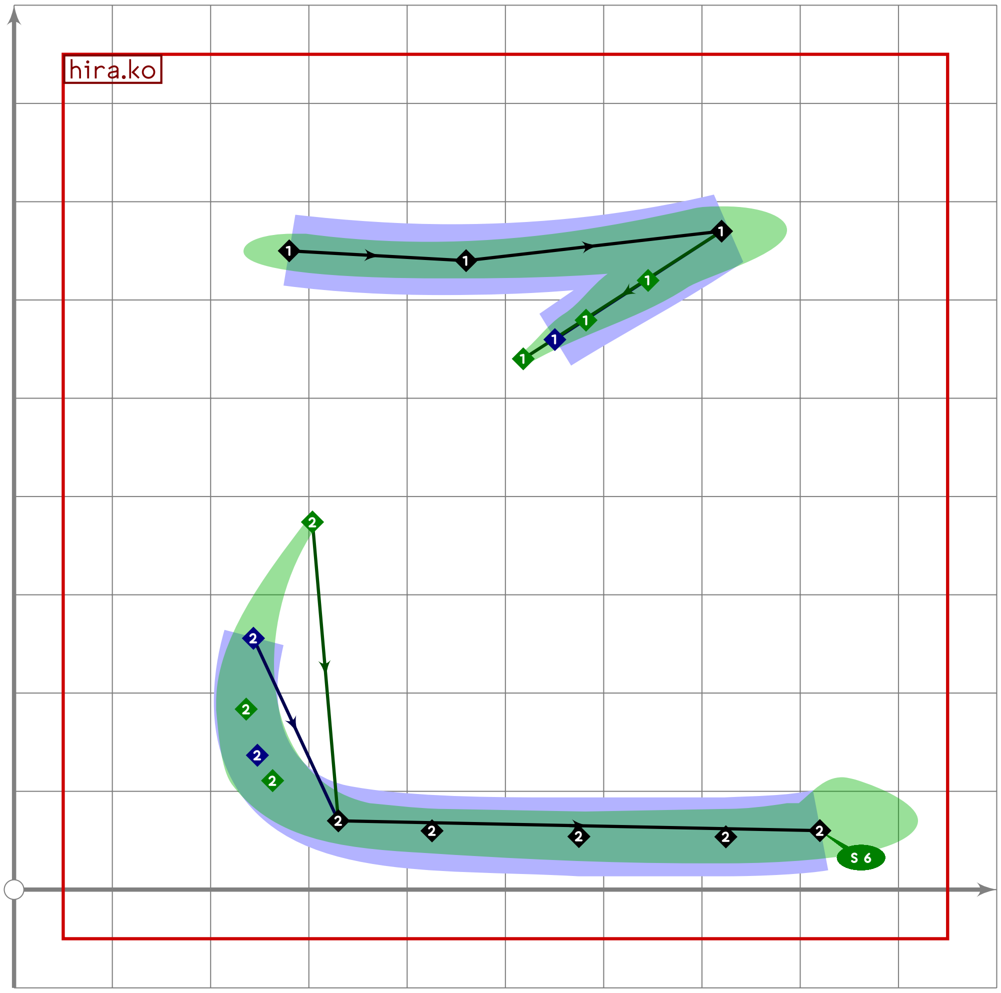


```

vardef hira.ku =
  push_pbox_toexpand("hira.ku");

  push_stroke((640,760)..(510,610)..(340,450)..
      tension 0.75..(340,370)..(530,190)..(670,10),
    (1.9,1.9)..(1.6,1.6)..(1.2,1.2)..(1.2,1.2)..(1.7,1.7)..(2.1,2.1));
  set_boserif(0,0,5);
  expand_pbox;
enddef;
```



```

vardef hira.ke =
  push_pbox_toexpand("hira.ke");

  hira.ni_left;

  push_stroke((390,500+15*mincho)..(470,485-5*mincho)..(600,490)..(880,540),
    (1,1)..(1.4,1.4)..(1.8,1.8)..(2,2));
  set_boserif(0,3,6);

  push_stroke((690,770)..tension 2..(700,210)..(280,-10),
    (1.6,1.6)--(1.4,1.4)--(0.6,0.6));
  set_boserif(0,0,8);
  expand_pbox;
enddef;
```

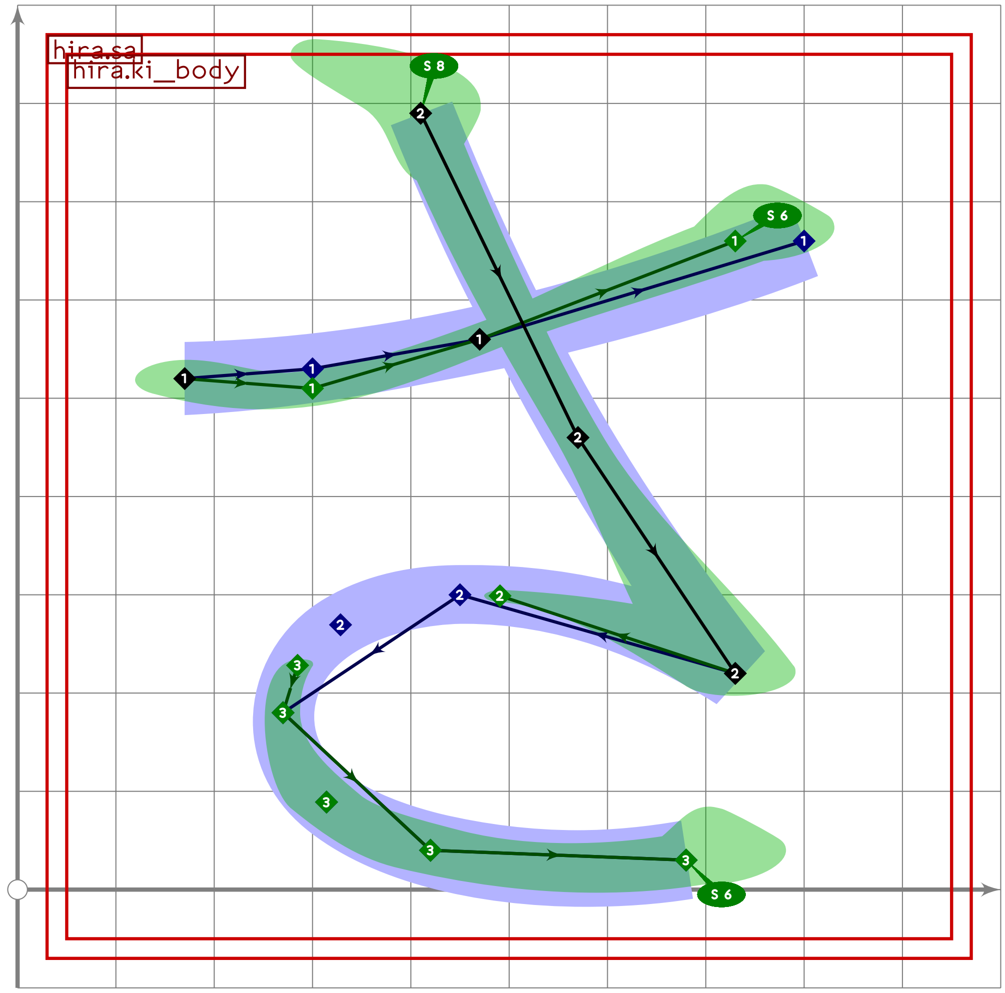
U+3053
tsuku.uni3053



```
vardef hira.ko =
  push_pbox_toexpand("hira.ko");

  push_stroke((280,650)..(460,640)..{curl 1}(720,670){curl 1}..
      (450,500)..(330,70)..tension 2.4..(820,60),
    (1.8,1.8)--(1.9,1.9)--(2.3,2.3)--
      (0.35,0.25)--(2.2,1.8)..(2.8,2.4));
  set_botip(0,2,0);
  set_boserif(0,5,6);
  expand_pbox;
enddef;

```

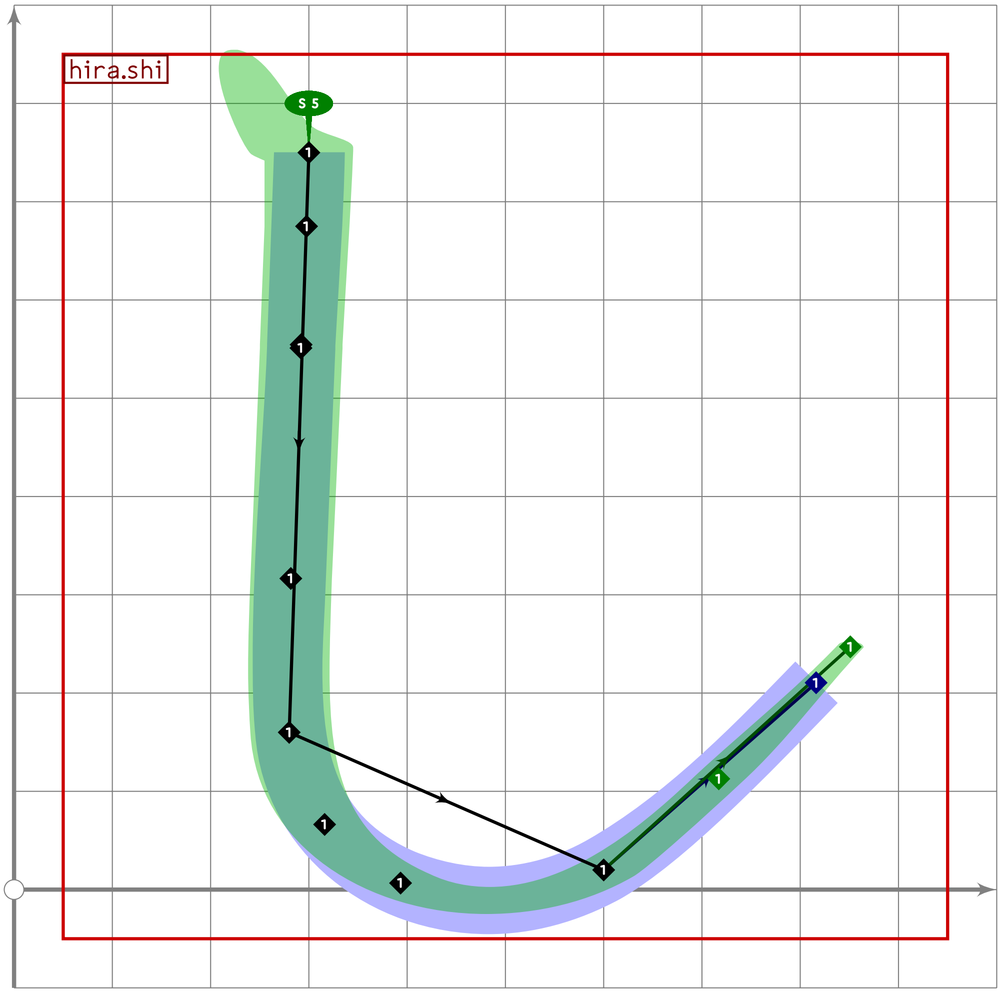
Hiragana Sashisuseso/Zajizuzezo

```
%%%%%%%%%% HIRAGANA SASHISUSESO/ZAJIZUZEZO
```



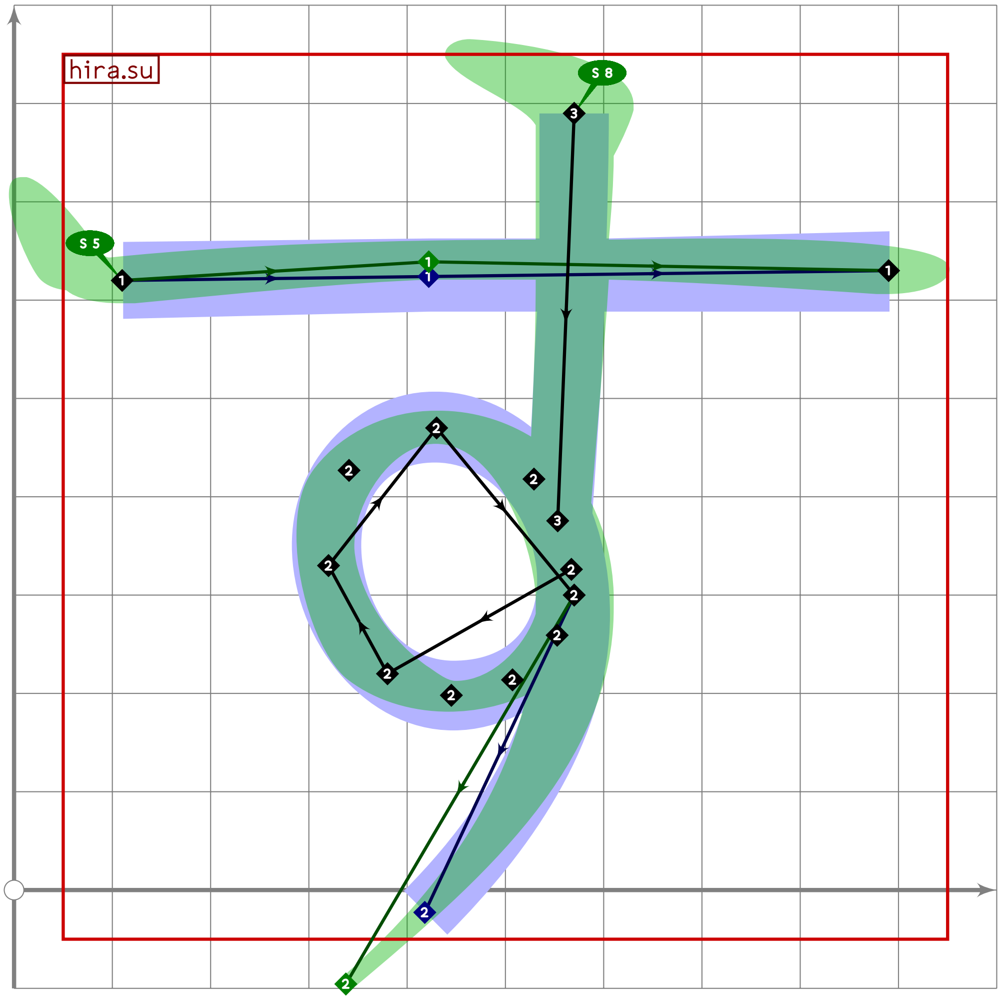
```

vardef hira.sa =
  push_pbox_toexpand("hira.sa");

  push_stroke((170,520)..(300,530-20*mincho)..(470,560)..(800-70*mincho,660),
    (1.9,1.9)--(1.8,1.8)--(1.4,1.4)--(2.1,2.1));
  set_boserif(0,3,6);

  hira.ki_body;
  expand_pbox;
enddef;
```

U+3057
tsuku.uni3057



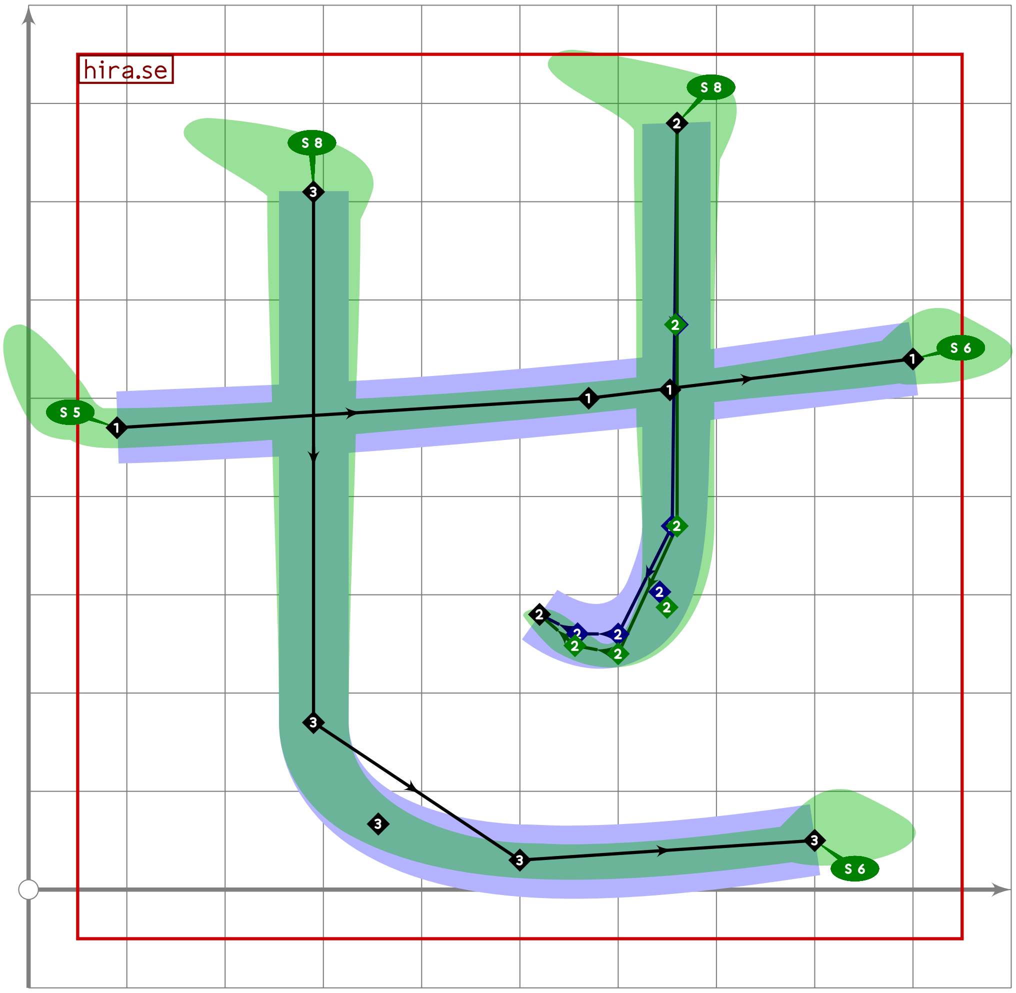
```

vardef hira.shi =
  push_pbox_toexpand("hira.shi");

  push_stroke((300,750){down}..tension 2.5..(280,160)..
       (600,20)..tension 1.5..{curl 0}(990,400),
    (1.7,1.7)..(1.6,1.6)--(1.5,1.5)--(0.4,0.55));
  set_boserif(0,0,5);
  expand_pbox;
enddef;
```

U+3059
tsuku.uni3059

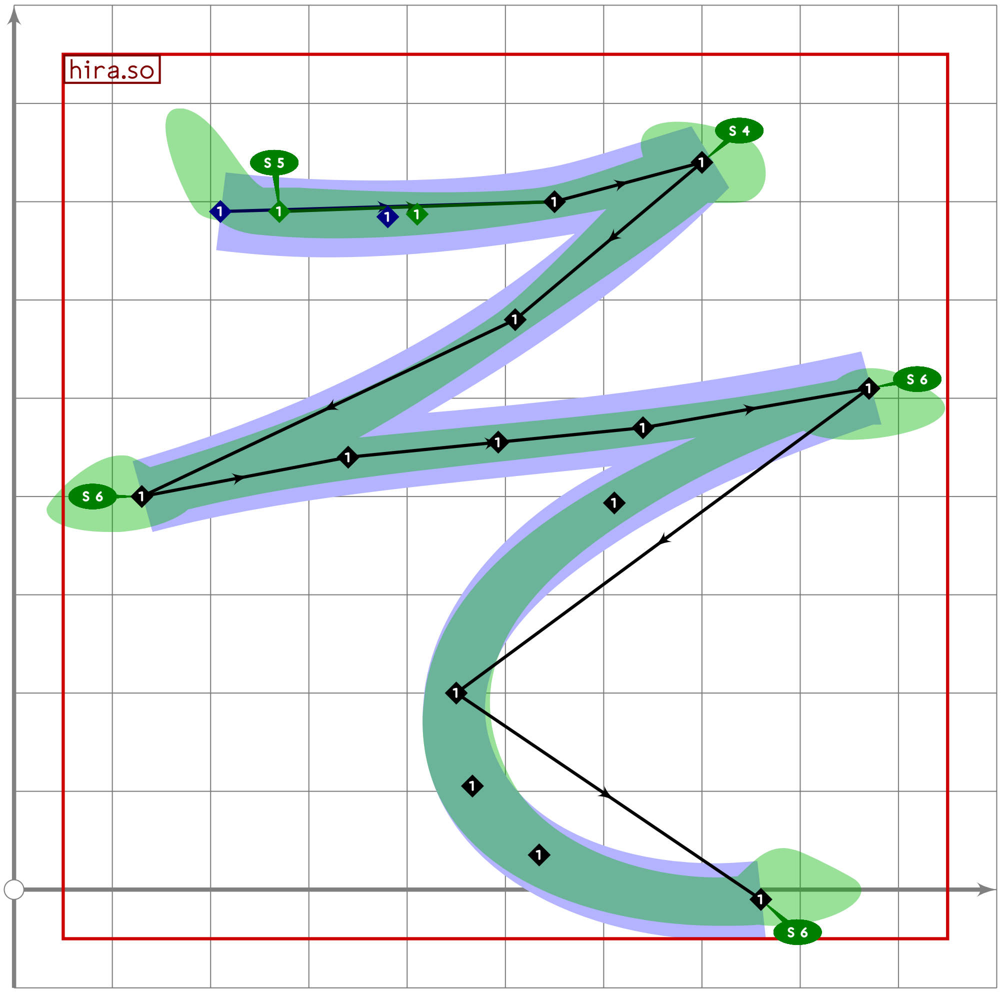


```
vardef hira.su =
  push_pbox_toexpand("hira.su");

  push_stroke((110,620)..(0.4[(110,620),(890,630)]+15*up*mincho)..(890,630),
    (2.2,2.2)--(1.9,1.9)--(2.2,2.2));
  set_boserif(0,0,5);

  push_stroke((320,330)..(430,470)..(570,300)..{curl 0}(270,-150),
    (1.3,1.3)--(1.7,1.7)--(1.3,1.3)--(1.7,1.7)--(1.6,1.6)--(0.7,0.7));
  replace_strokep(0)((point 1.9 of oldp)..(380,220)..oldp);

  push_stroke((570,790){down}..(point 3.7 of get_strokep(0)),
    (1.6,1.6)..(1.4,1.4));
  set_boserif(0,0,8);
  expand_pbox;
enddef;
```

U+305B
tsuku.uni305B



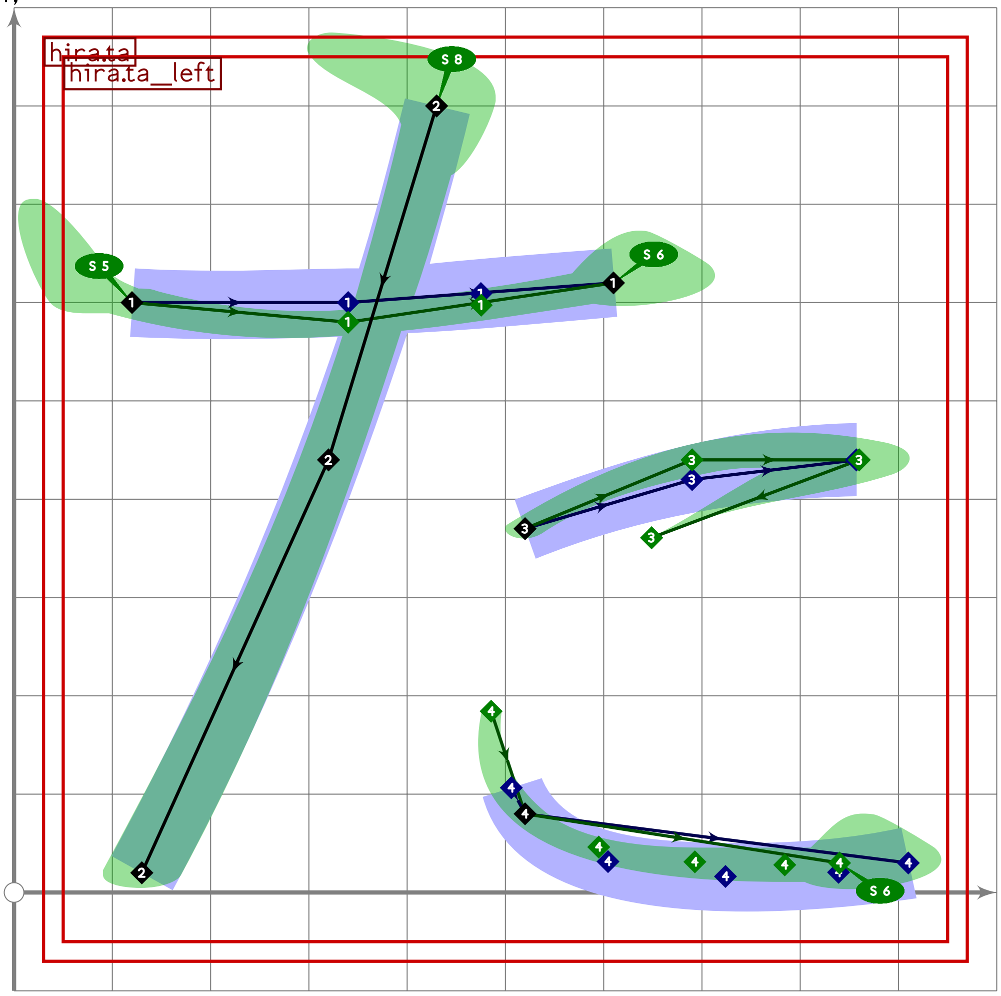
```
vardef hira.se =
  push_pbox_toexpand("hira.se");

  push_stroke((90,470)..(570,500)..(900,540),
    (2,2)..(1.6,1.6)..(2,2));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  push_stroke(insert_nodes((660,780)..tension 1.5..(655+5*mincho,370)..
      (600,260-20*mincho)..{curl 0.2}(520,280))(2.5),
    (1.7,1.7)--(1.5,1.5)--(1.3,1.3)--(1.2,1.2)--(1,1));
  set_boserif(0,0,8);

  push_stroke((290,710)..(290,170){down}..(500,30)..
      {direction infinity of get_strokep(-1)}(800,50),
    (1.8,1.8)--(1.5,1.5)--(1.7,1.7)--(1.8,1.8));
  set_boserif(0,0,8);
  set_boserif(0,3,6);
  expand_pbox;
enddef;
```



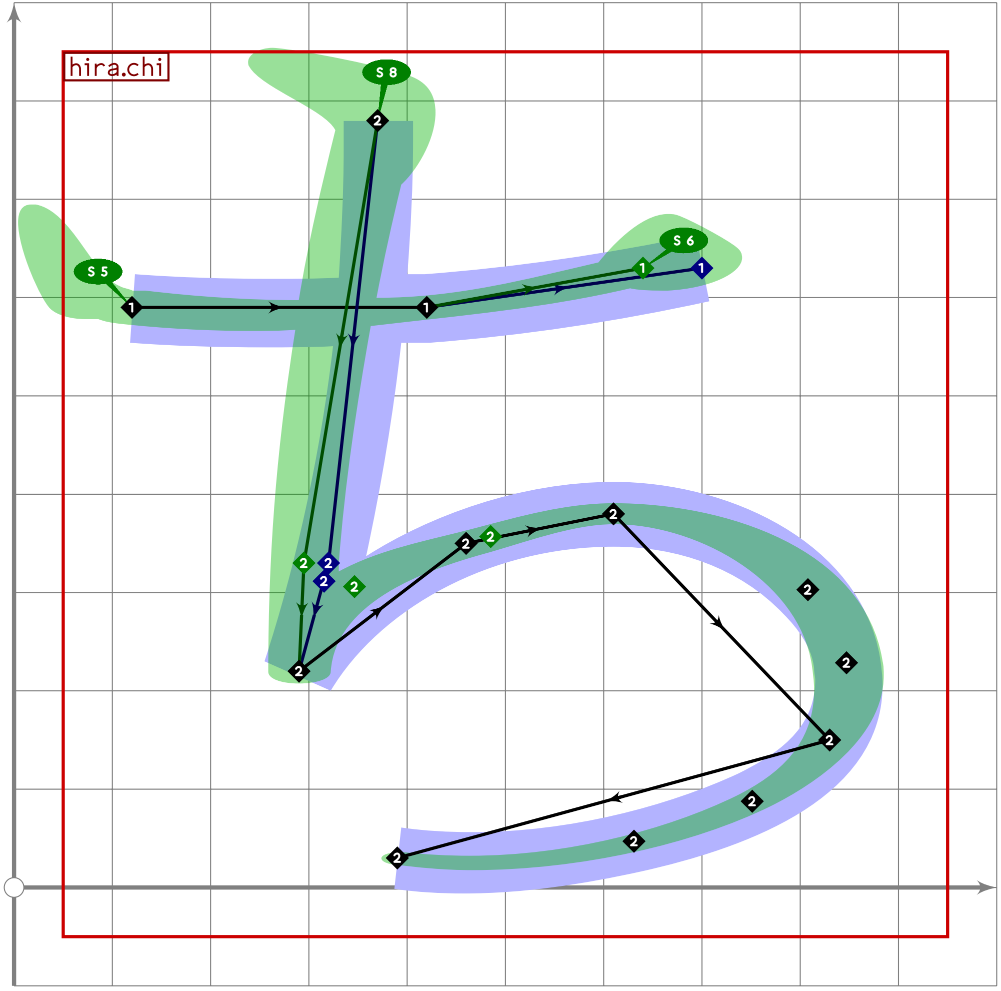
```
vardef hira.so =
  push_pbox_toexpand("hira.so");

  push_stroke(
    (210+60*mincho,690)..tension 1.2..(550,700)..
      {curl 0}(700,740){curl 1}..
      (510,580)..
      {curl 1}(130,400){curl 2}..(340,440)..(640,470)..
      {curl 1}(870,510){curl 0}..tension 1.2..(450,200)..{curl 0.2}(760,-10),
    (2.3,2.3)--(1.7,1.7)--(1.8,1.8)--
      (1.2,1.2)--
      (2.1,2.1)--(1.9,1.9)--(1.7,1.7)--(1.5,1.5)--
      (1.4,1.4)--(2.3,2.3));
  set_botip(0,2,0);
  set_botip(0,4,0);
  set_botip(0,7,0);
  set_boserif(0,0,5);
  set_boserif(0,2,4);
  set_boserif(0,4,6);
  set_boserif(0,7,6);
  set_boserif(0,9,6);
  expand_pbox;
enddef;

```

Hiragana Tachitsuteto/Dajizudedo

```
%%%%%%%%%% HIRAGANA TACHITSUTETO/DAJIZUDEDO

vardef hira.ta_left =
  push_pbox_toexpand("hira.ta_left");

  push_stroke((120,600)..(340,600-20*mincho)..tension 1.5..(610,620),
    (1.6,1.6)..(1.5,1.5)..(1.7,1.7));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  push_stroke((430,800)..(320,440)..(130,20),
    (1.4,1.4)..(1.3,1.3)..(1.6,1.6));
  set_boserif(0,0,8);
  expand_pbox;
enddef;
```

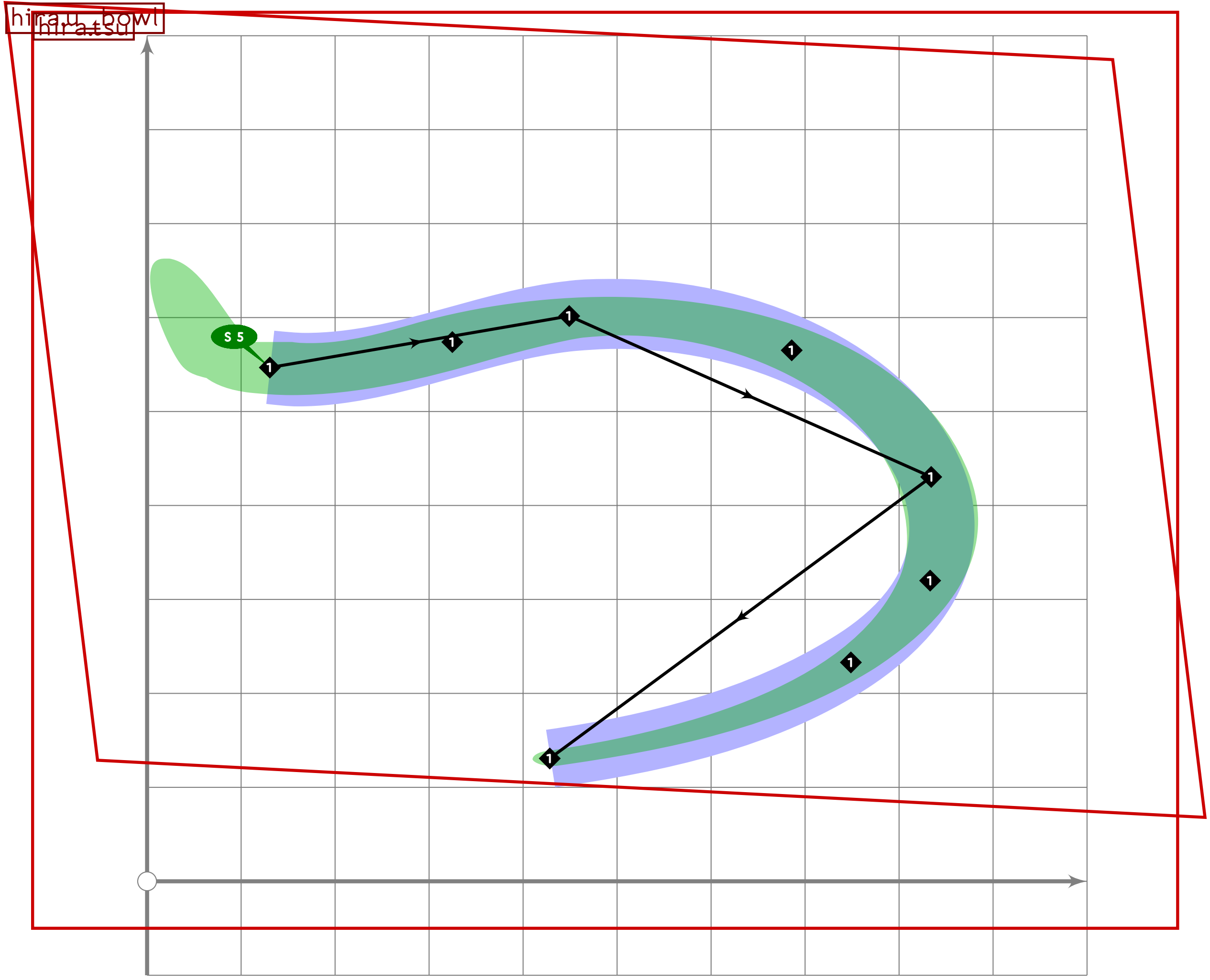


```

vardef hira.ta =
  push_pbox_toexpand(”hira.ta”);

  hira.ta_left;

  push_stroke((520,370)..(690,420+20*mincho)..{curl 1.5}(860,440){curl 0}..
      (610,280+60*mincho)..(520,80)..tension 1.2 and 3..
      {curl 0.2}(910-70*mincho,30),
    (1.1,1.1)--(1.6,1.6)--(2.8,0.99)--(0.45,0.35)--(1.1,1.1)--(1.9,1.9));
  set_botip(0,2,0);
  set_boserif(0,5,6);
  expand_pbox;
enddef;

vardef hira.chi_bottom =
  replace_strokep(0)((oldp){-direction infinity of oldp xscaled 2}..
    (460,350)..(610,380){right}..(830,150)..
    tension 1.4..{curl 0.3}(390,30));
  replace_strokeq(0)((oldq)..(1.3,1.3)..(1.5,1.5)..(1.5,1.5)..(1,1));
enddef;
```

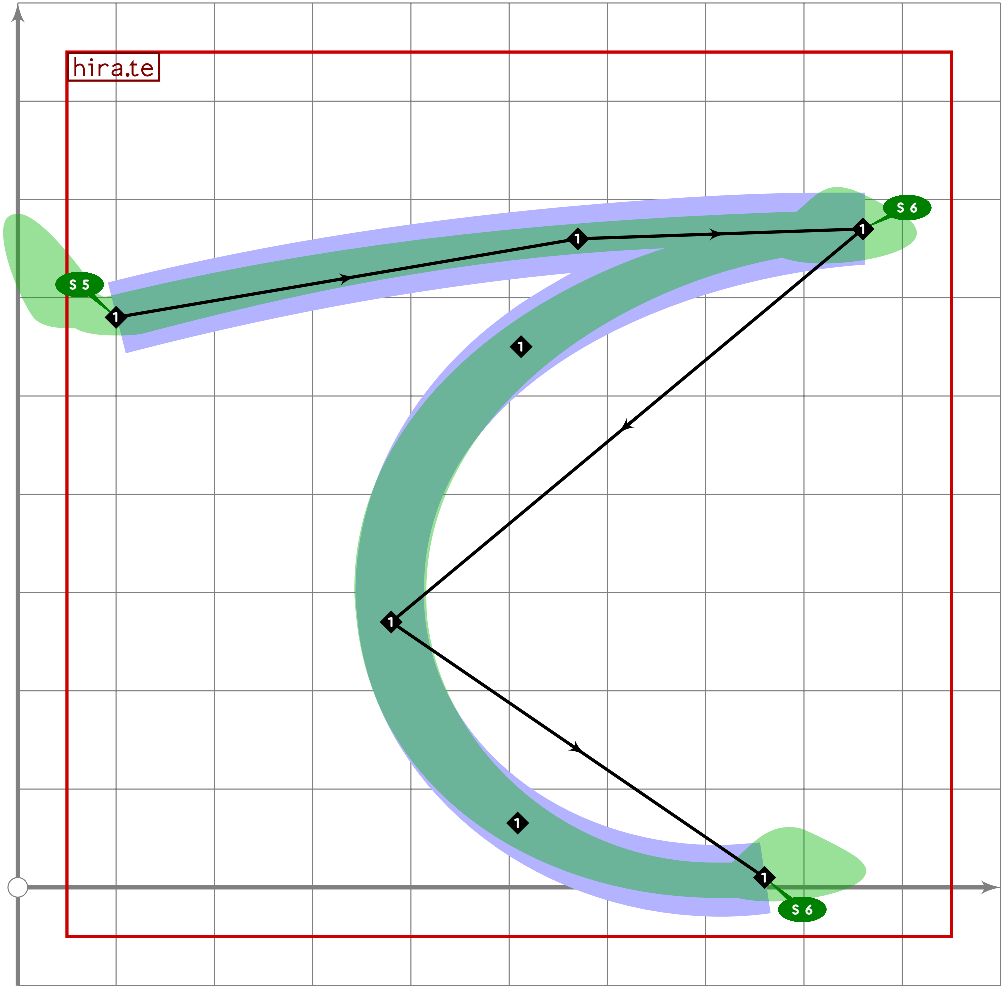


U+3064
tsuku.uni3064

```

vardef  hira.chi =
  push_pbox_toexpand("hira.chi");

  push_stroke((120,590)..(420,590)..(700-60*mincho,630),
    (1.7,1.7)..(1.5,1.5)..(1.6,1.6));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  push_stroke((370,780)..(320-25*mincho,330)..(290,220),
    (1.6,1.6)..(1.4,1.4)..(1.5,1.5));
  hira.chi_bottom;
  set_botip(0,2,0);
  set_boserif(0,0,8);
  expand_pbox;
enddef;
```



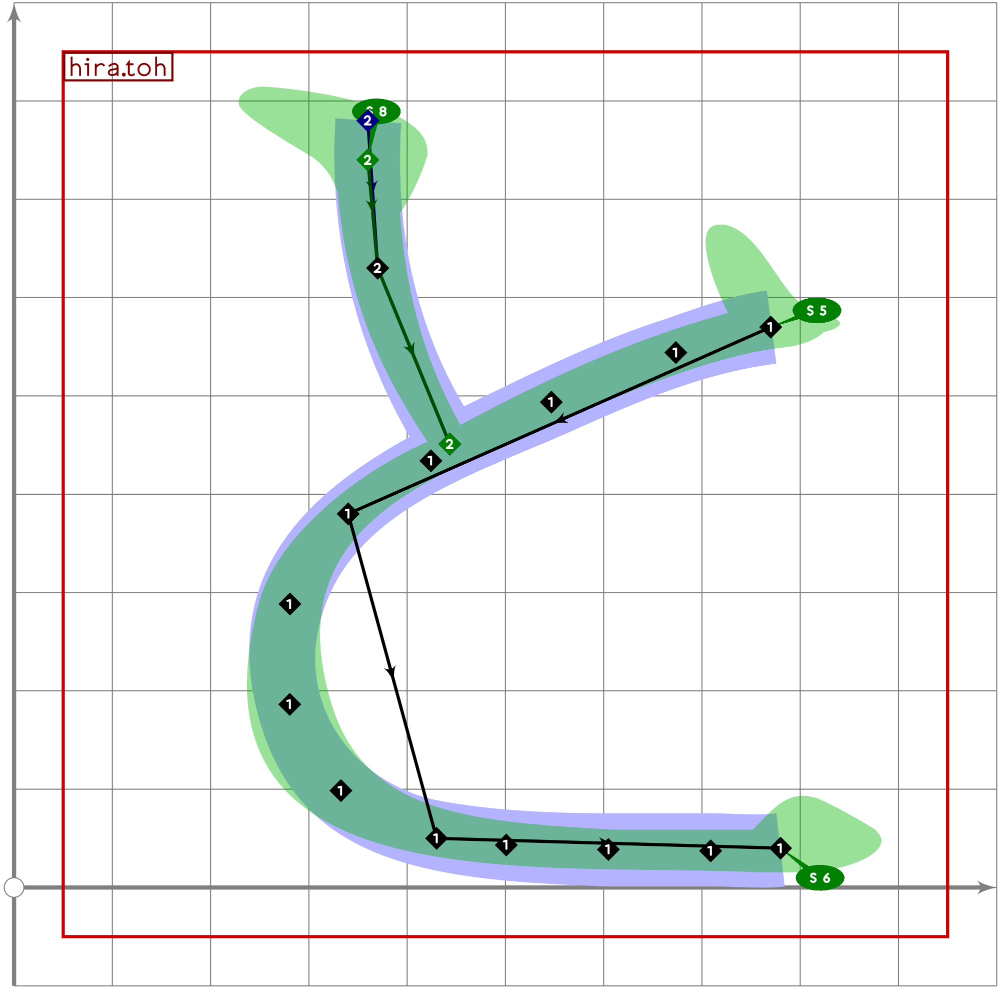
```

vardef  hira.tsu =
  push_pbox_toexpand("hira.tsu");

  begingroup
    save xf;
    transform xf;
    (300,450) transformed xf=(220,560);
    (750,350) transformed xf=(820,440);
    (400,0) transformed xf=(400,150);
    tsu_xform(xf)(hira.u_bowl);
    set_bosize(0)(100+10*mincho);
  endgroup;
  expand_pbox;
enddef;
```



```

vardef hira.te =
  push_pbox_toexpand("hira.te");

  push_stroke((100,580)..(570,660)..{curl 1}(860,670){curl 0.2}..
    (380,270)..{curl 0.6}(760,10),
    (1.9,1.9)--(1.5,1.5)--(1.8,1.8)--(1.5,1.5)--(1.8,1.8));
  set_botip(0,2,0);
  set_boserif(0,0,5);
  set_boserif(0,2,6);
```

U+3068
tsuku.uni3068

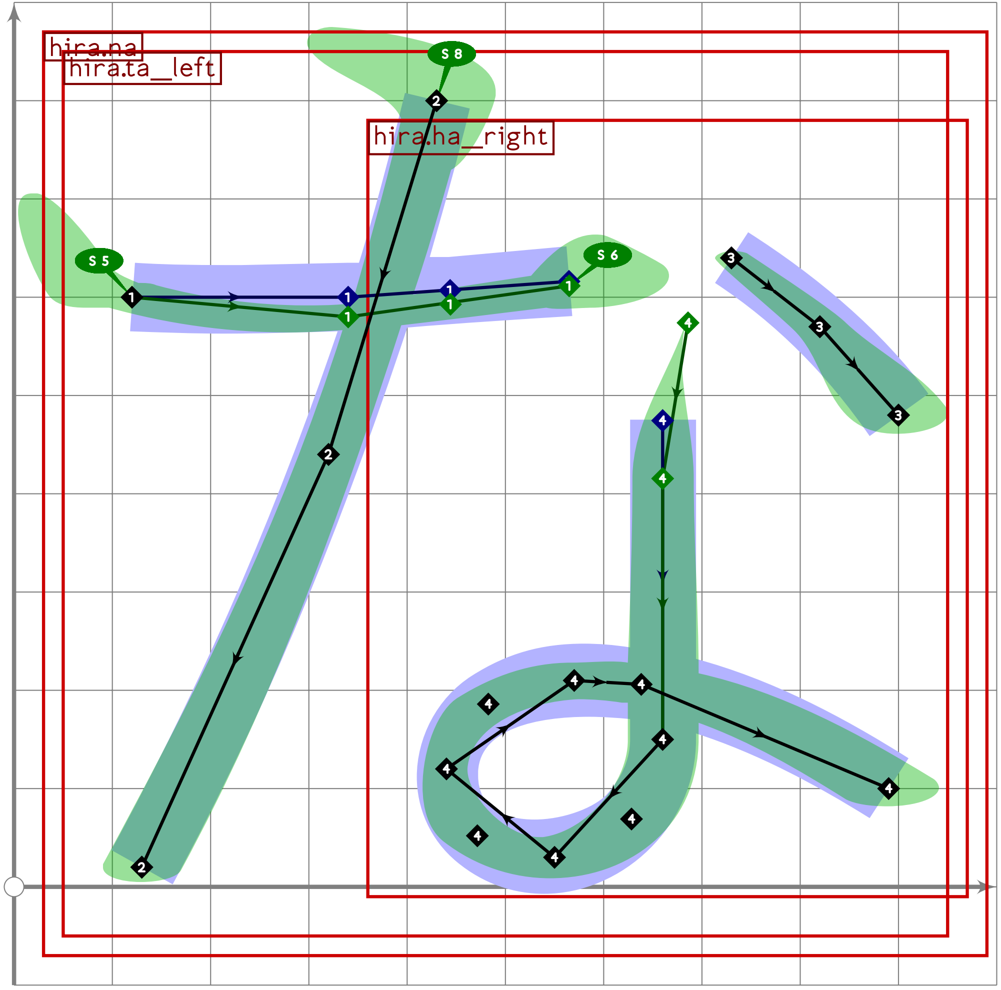
```
  set_boserif(0,4,6);
  expand_pbox;
enddef;
```



```

vardef hira.toh =
  push_pbox_toexpand(”hira.toh”);

  push_stroke((770,570)..tension 1.7..(340,380)..(430,50)..
      tension 2..(780,40),
    (2,2)–(1.2,1.2)–(2.1,2.1)–(2,2));
  set_boserif(0,0,5);
  set_boserif(0,3,6);

  push_stroke(subpath (0,1.97) of
      ((360,780-40*mincho)..(370,630)..(point 0.7 of get_strokep(0))),
    (1.4,1.4)–(1.3,1.3)–(1.1,1.1));
  set_boserif(0,0,8);
  expand_pbox;
enddef;

```

Hiragana Naninuneno

```
%%%%%%%%%% HIRAGANA NANINUNENO
```

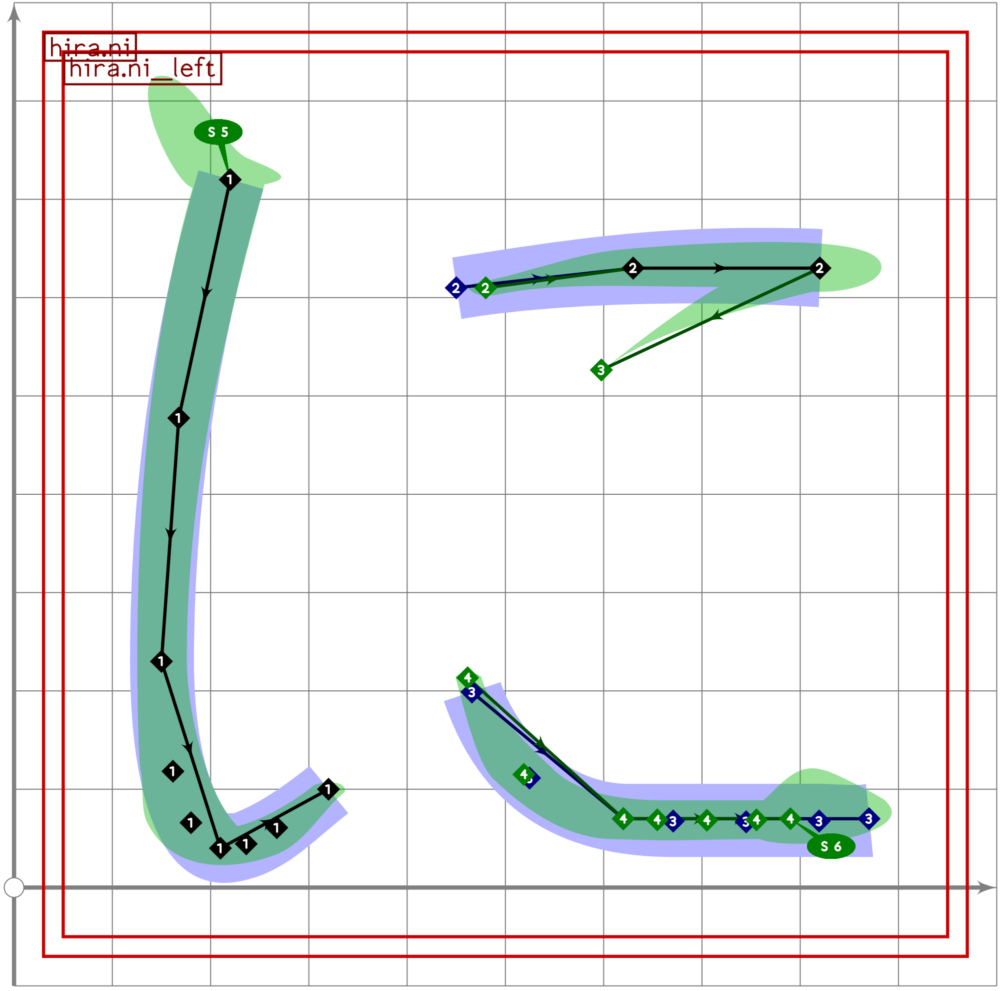


```

vardef hira.na =
  push_pbox_toexpand("hira.na");

  hira.ta_left;
  replace_strokep(-1)(subpath (0,1.8) of oldp);

  push_stroke((730,640)..(820,570)..(900,480),
    (1,1)..(1.3,1.3)..(1.9,1.9));

  hira.ha_right;

  replace_strokep(0)(((120,0)+point 0 of oldp)
    ..(subpath (0.45+0.1*mincho,infinity) of oldp));
  replace_strokeq(0)((-0.2,-0.4)--(1.7,1)--(1.5,1.5)--(2,2)--
    (1.1,1.1)--(1.9,1.9));
  expand_pbox;
enddef;

```

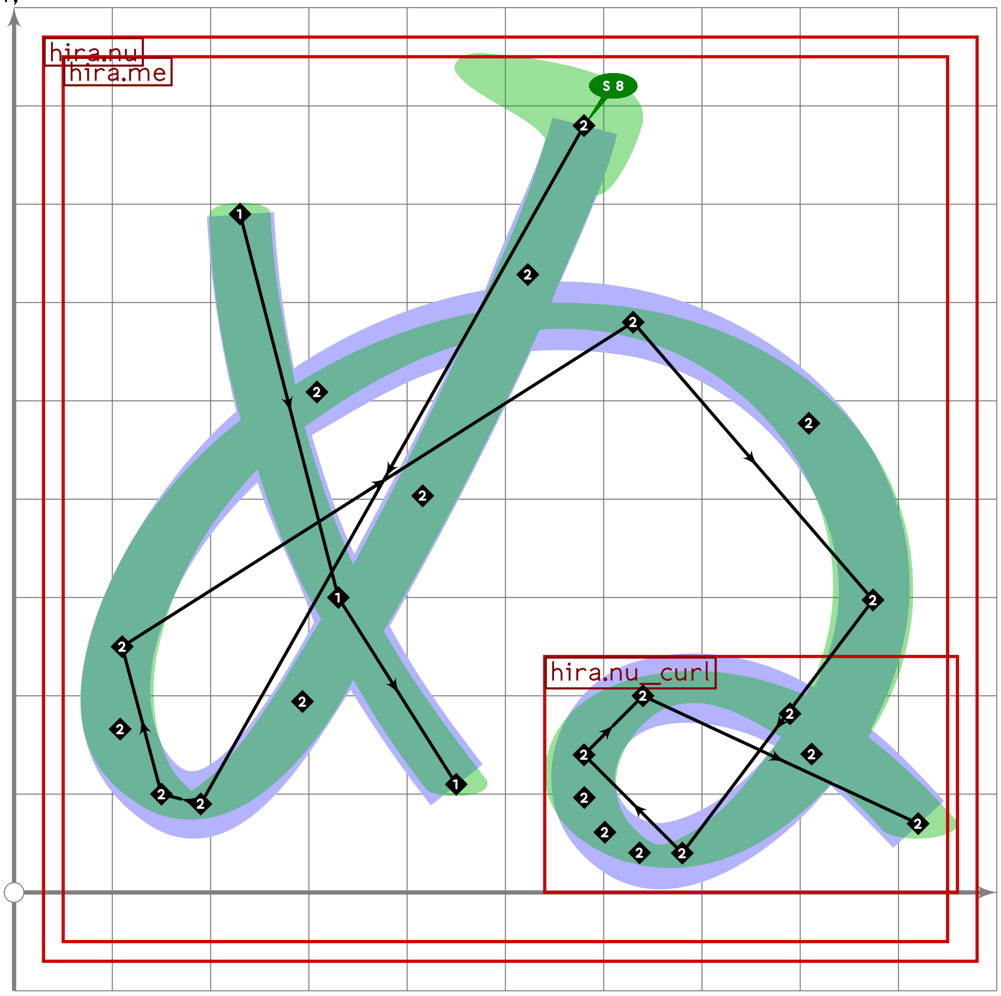
HIRA

U+306B
tsuku.uni306B

```
vardef hira.ni_left =
  push_pbox_toexpand("hira.ni_left");

  push_stroke((220,720)..(150,230){down}..tension 1.5..(210,40)..
      tension 1.5..{curl 0}(320,100),
    (1.5,1.5)..(1.2,1.2)..(1.8,1.8)..(1,1));
  replace_strokep(0)(insert_nodes(oldp)(0.5));
  replace_strokeq(0)(insert_nodes(oldq)(0.5));
  set_boserif(0,0,5);
  expand_pbox;
enddef;
```



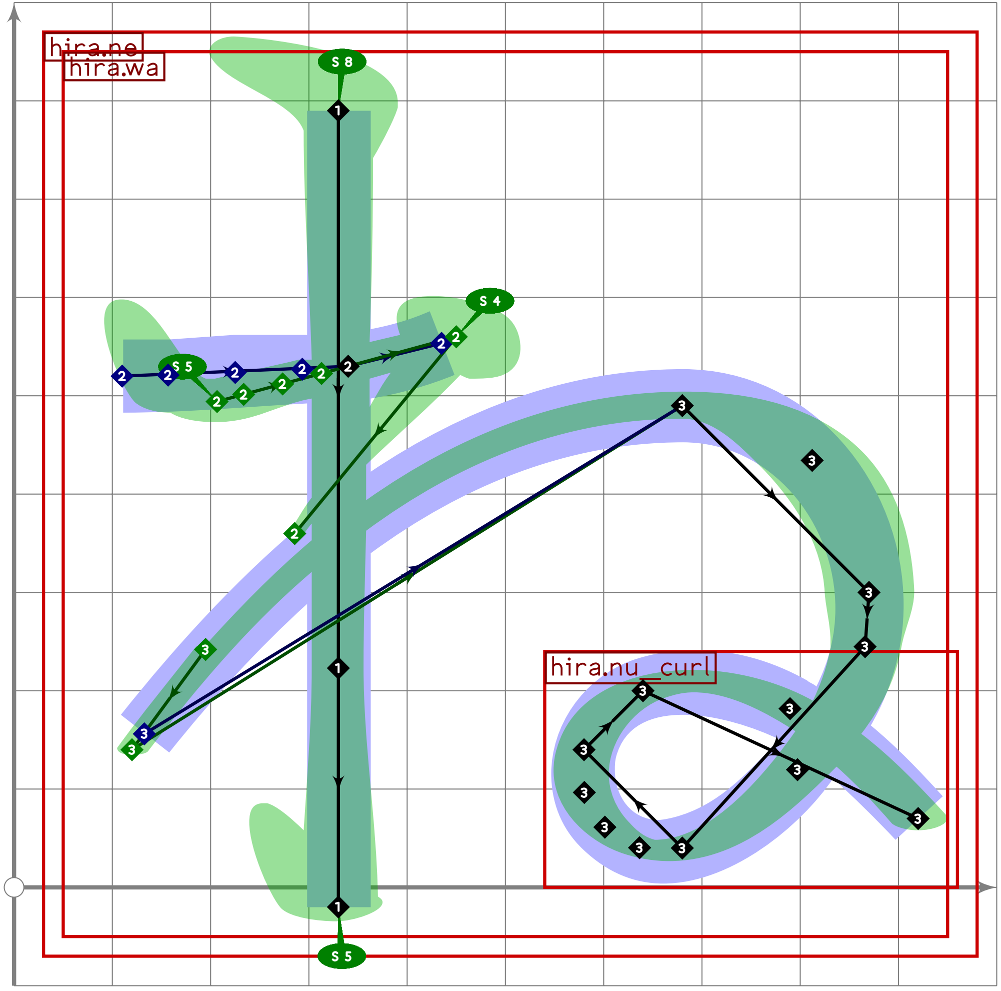
```

vardef hira.ni =
  push_pbox_toexpand("hira.ni");

  hira.ni_left;

  push_stroke((450+30*mincho,610)..(630,630)..(820,630),
    (1,1)..(1.9,1.9)..(2.2,2.2));

  push_stroke((point infinity of get_strokep(0)){curl 0.6}..(530,460)..
      (460,220)..(620,70)..tension 2..(870-80*mincho,70),
    (3,0.7)--(0,0)--(0.8,0.9)--(2,2)--(1.9,1.9));
  set_boserif(0,4,6);
  expand_pbox;
enddef;

vardef hira.nu_curl =
  begingroup
    push_pbox_explicit("hira.nu_curl",
      identity xyscaled (420,240) shifted (540,0));
    (680,40)..(580,140)..(640,200)..{curl 0}(920,70)
  endgroup
enddef;
```



```

vardef hira.nu =
  push_pbox_toexpand("hira.nu");

  hira.me;

  replace_strokep(0)((subpath (0,4.8) of oldp)..tension 1.2..
    hira.nu_curl);
```

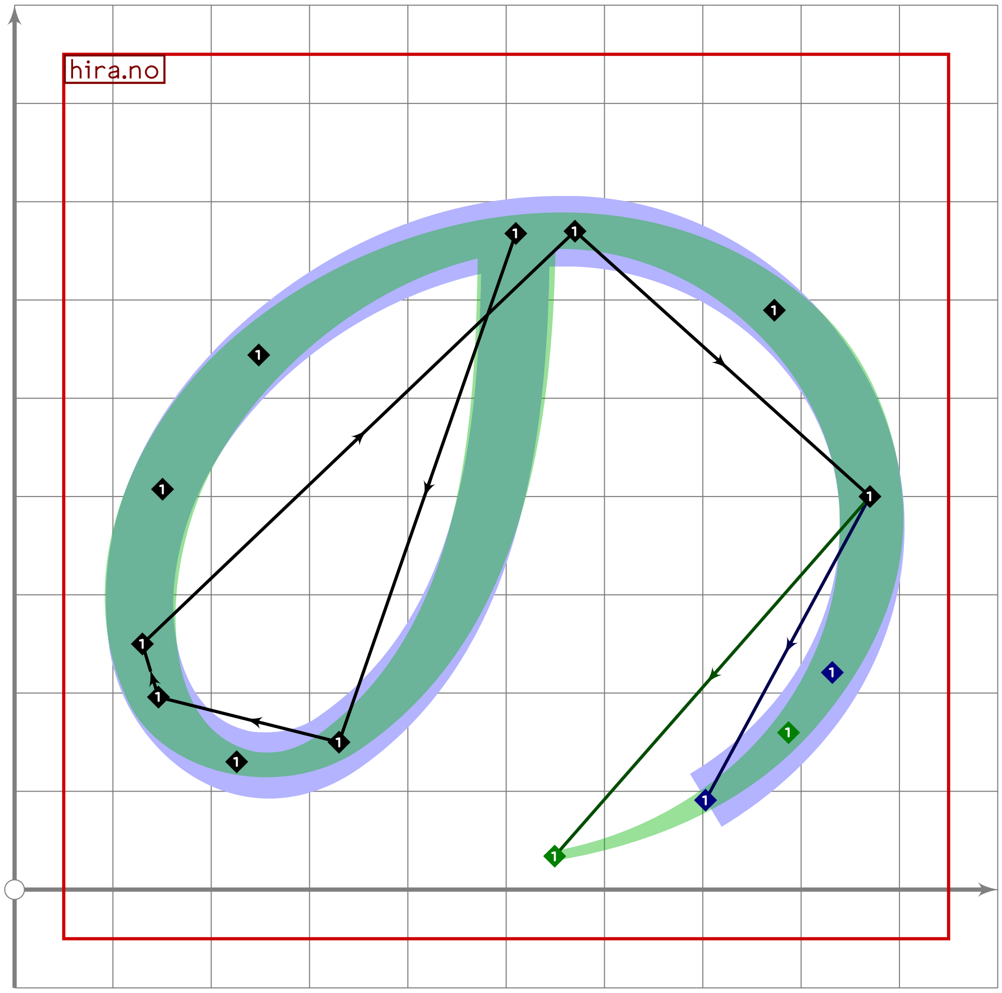
U+306D
tsuku.uni306D

```
  replace_strokeq(0)((1.5,1.5)--(1.4,1.4)--(1.6,1.6)--(1.4,1.4)--
    (1.6,1.6)--(1.6,1.6)--(1.7,1.7)--(1.3,1.3)--(1.6,1.6));
  set_boserif(0,0,8);
  expand_pbox;
enddef;
```



```

vardef hira.ne =
  push_pbox_toexpand("hira.ne");

  hira.wa;

  replace_strokep(0)((subpath (0,6.1) of oldp)..tension 1.2..
    hira.nu_curl);
  replace_strokeq(0)((2,2)--(1.6,1.6)--(2.2,0.9)--(0.7,0.7)--(0.97,0.97)--
    (2,2)--(1.5,1.5)--(1.4,1.4)--(1.4,1.4)--(1.2,1.2)--(1.3,1.3));
  expand_pbox;
enddef;
```

U+306E
tsuku.uni306E

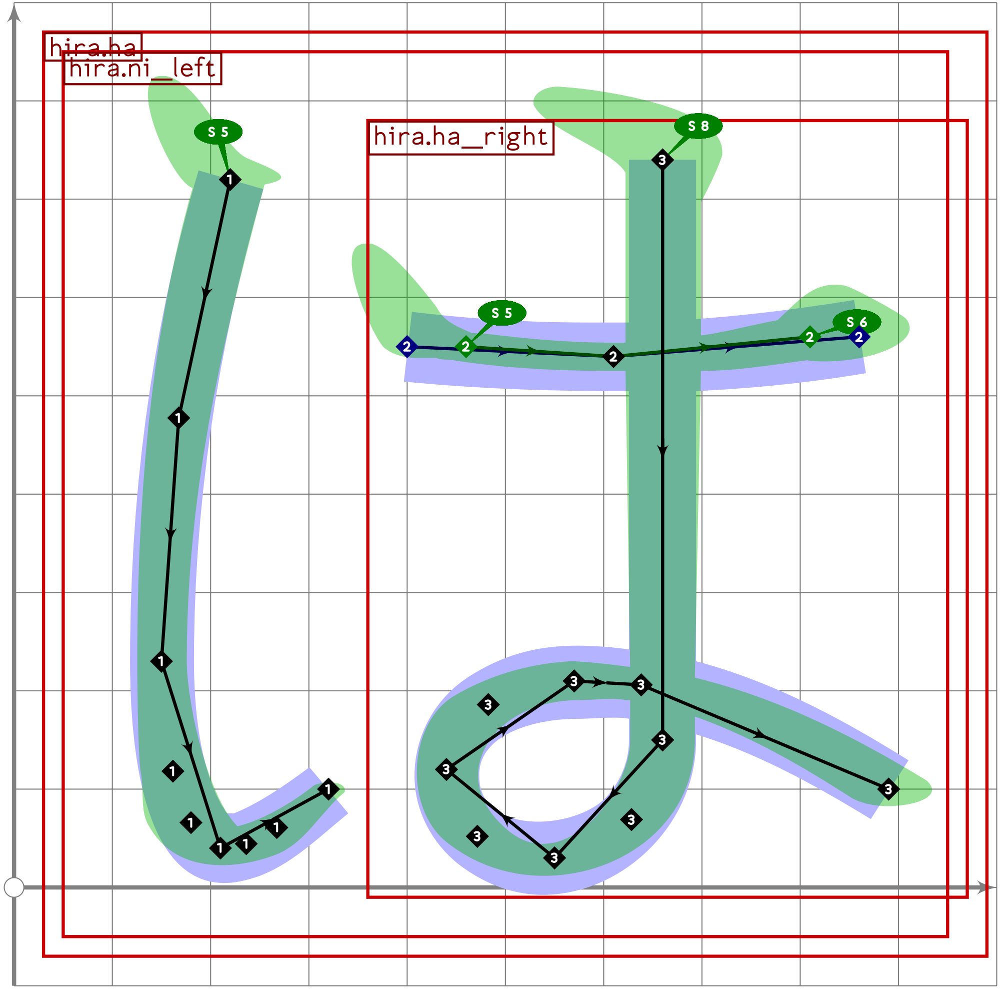


```
vardef hira.no =
  push_pbox_toexpand(”hira.no”);

  begingroup
    save px,py;
    path px,py;
    px:=(410,30)..(130,250)..tension 1.1..(570,670)..(870,400)..cycle;
    py:=(510,770){down}..{dir 215}(330,150);

    px:=subpath (0.85,4) of px;
    push_stroke(
        (subpath (xpart (py intersectiontimes px),infinity) of py)..px,
      (1.6,1.6)--(1.3,1.3)--(1.4,1.4)--(1.5,1.5)--(1.8,1.8)--
        (1.4,1.4)--(0.7,0.7));
  endgroup;
  expand_pbox;
enddef;
```

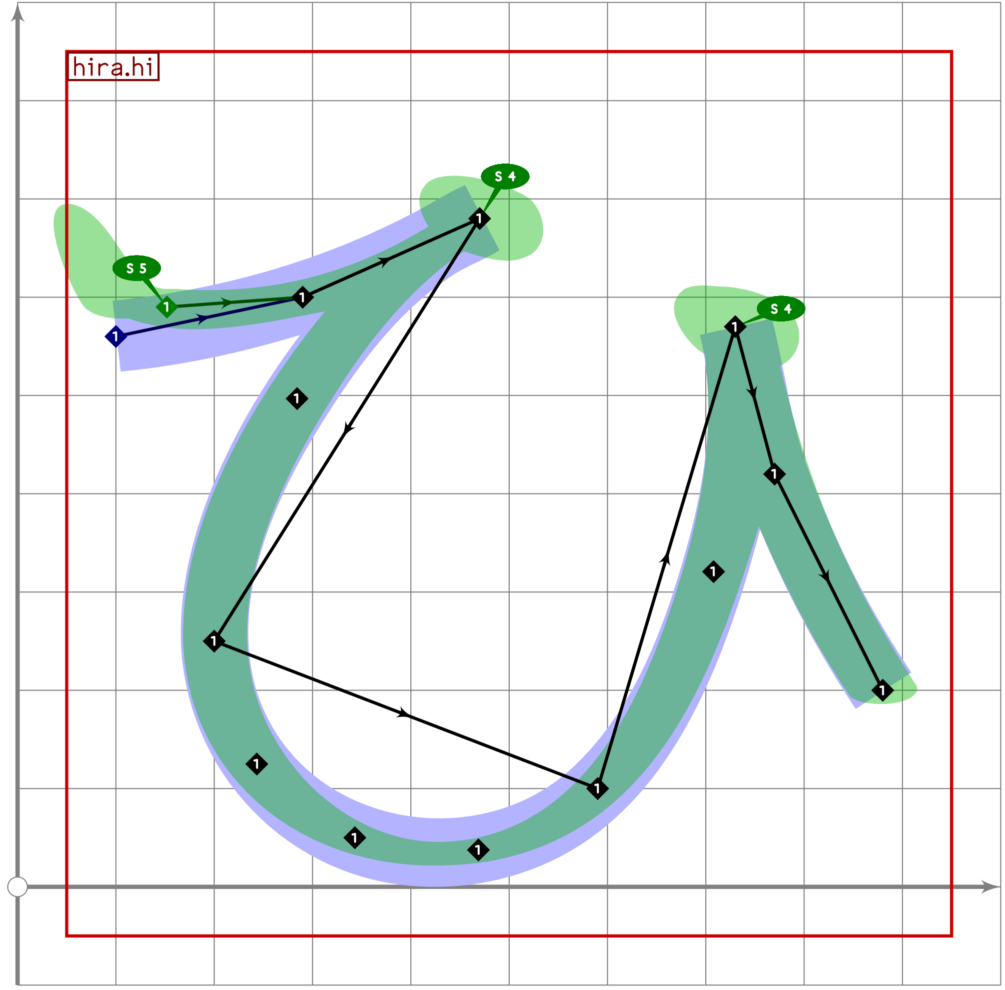
U+306F
tsuku.uni306F

Hiragana Hahifuheho/Babibubebo/Papipupepo

```
%%%%%%%%%% HIRAGANA HAHIFUHEHO/BABIBUBEBO/PAPIPUPEPO

vardef hira.ha_right =
  push_pbox_explicit("hira.ha_right",
    identity xyscaled (610,790) shifted (360,-10));

  push_stroke(insert_nodes((660,740)..(660,150){down}..(550,30)..(440,120)..
      (570,210)..{curl 0.17}(890,100))(4.2),
    (1.6,1.6)--(1.4,1.4)--(1.6,1.6)--(1.3,1.3)--
      (2,2)--(1.6,1.6)--(1.7,1.7));
enddef;
```



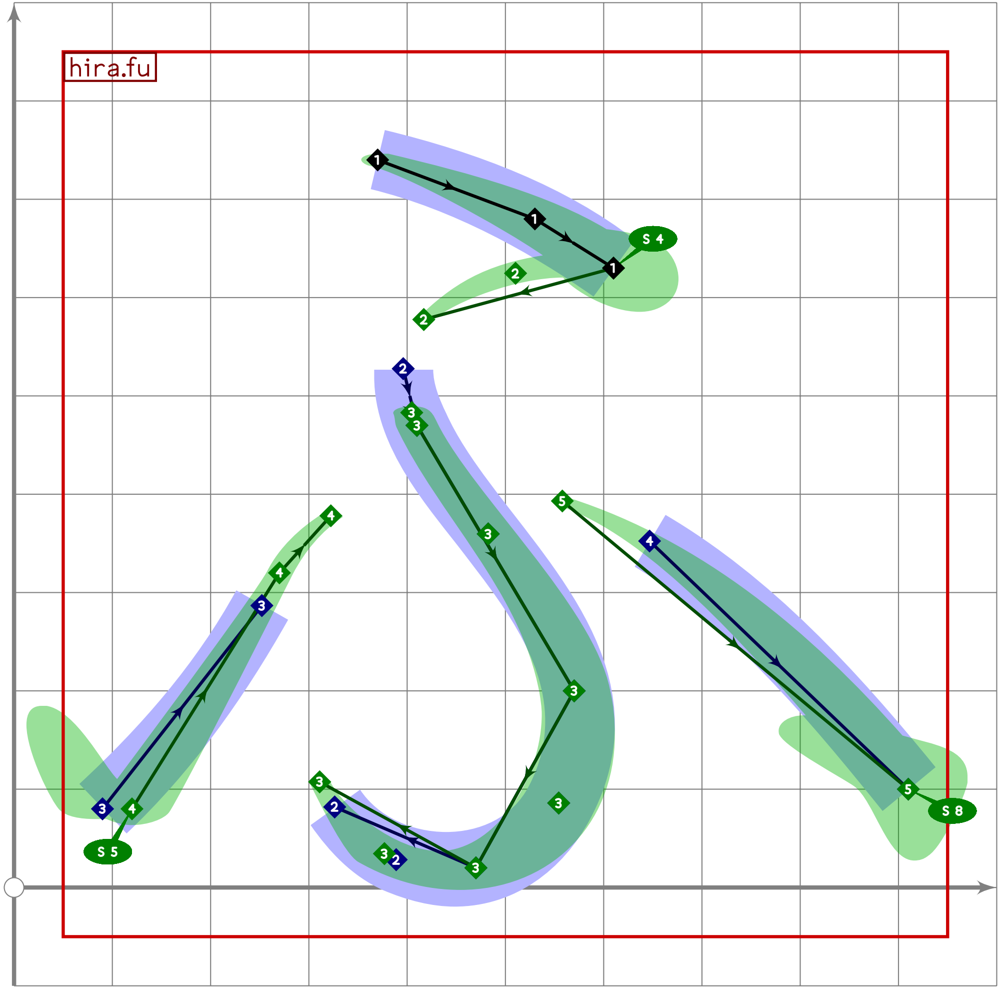
```

vardef hira.ha =
  push_pbox_toexpand("hira.ha");

  hira.ni_left;

  push_stroke((400+60*mincho,550)..(610,540)..(860-50*mincho,560),
    (1.6,1.6)..(1.6,1.6)..(2,2));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  hira.ha_right;
  set_boserif(0,0,8);
  expand_pbox;
enddef;
```



```

vardef hira.hi =
  push_pbox_toexpand("hira.hi");

  push_stroke(((100,560)+60*mincho*dir 30)..(290,600)..
      {curl 1}(470,680){curl 1}..
      tension 1.3..(200,250)..(590,100)..tension 1.3..
      {curl 1}(730,570){curl 1}..(770,420)..(880,200),
    (1.8,1.8)--(1.7,1.7)--(1.5,1.5)--
      (1.4,1.4)--(1.4,1.4)--(1.4,1.4)--(1.3,1.3)--(1.5,1.5));
  set_botip(0,2,0);
  set_botip(0,5,0);
  set_boserif(0,0,5);
  set_boserif(0,2,4);
```

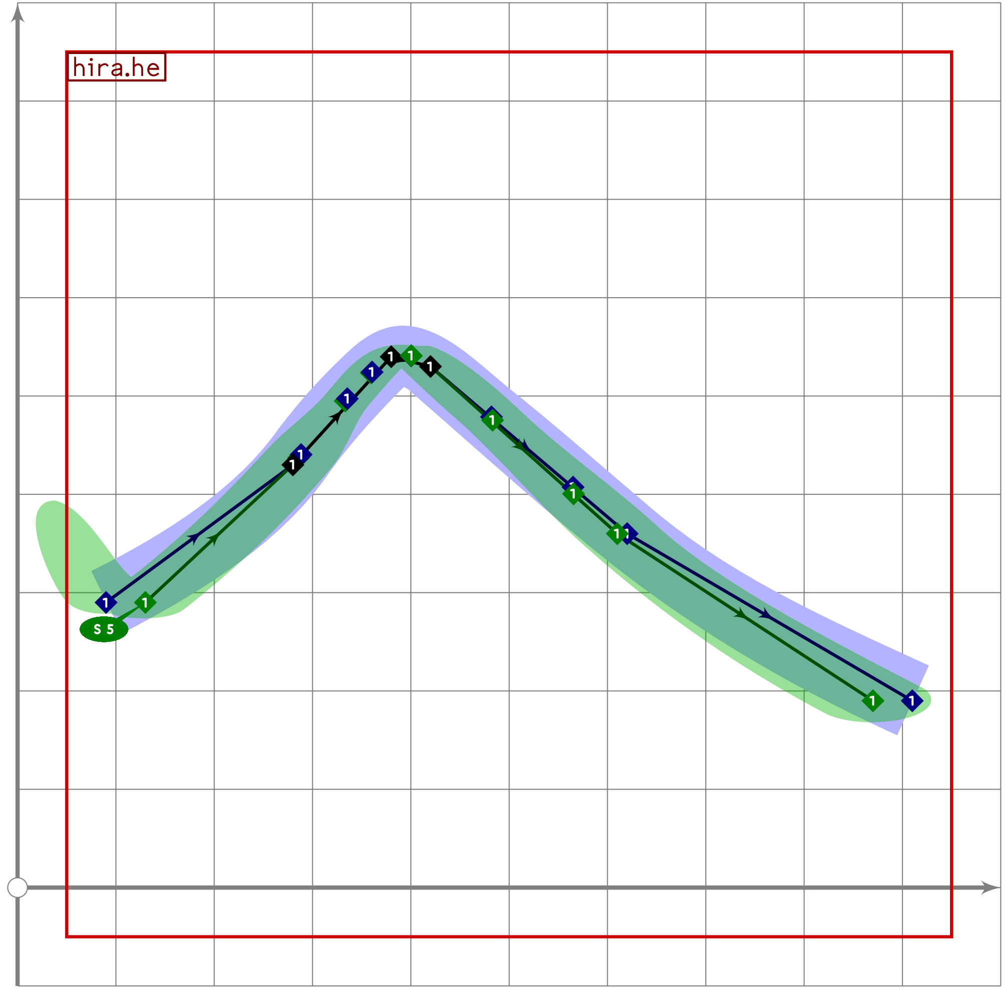
U+3075
tsuku.uni3075

```
  set_boserif(0,5,4);
  expand_pbox;
enddef;
```



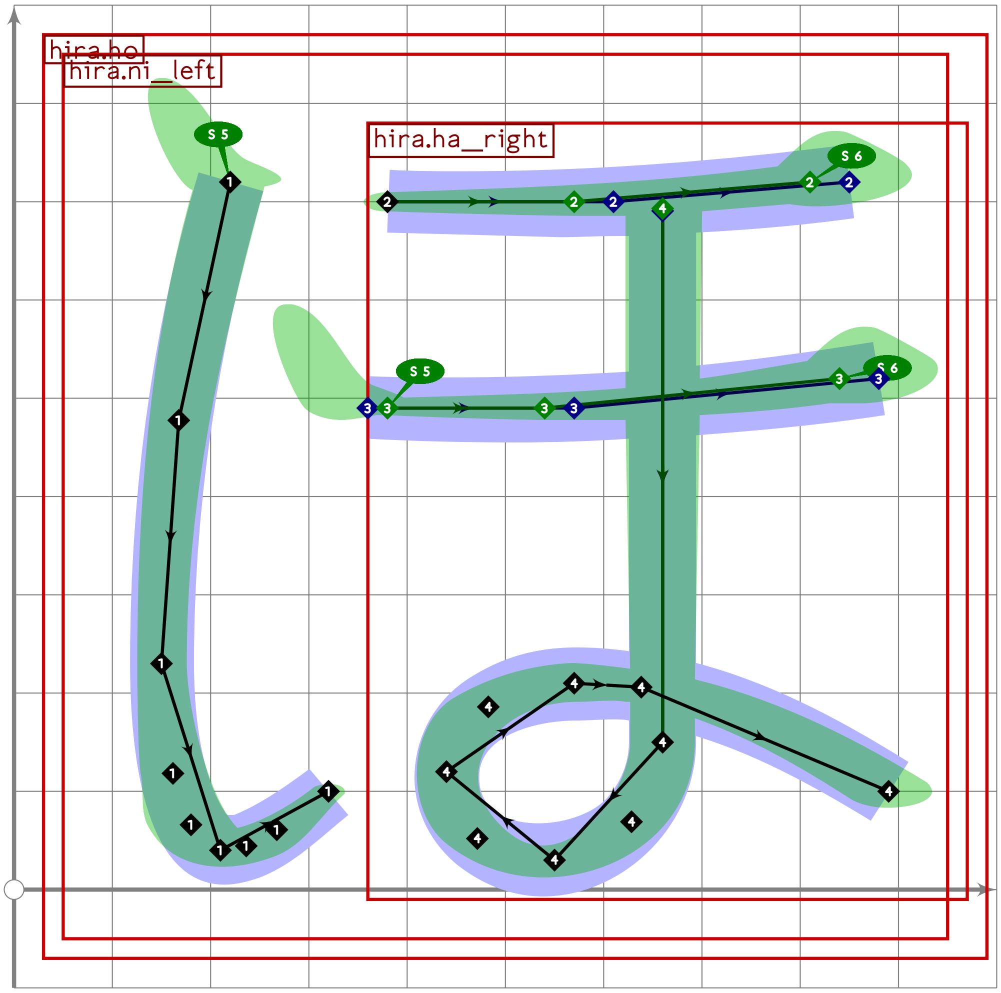
```

vardef hira.fu =
  push_pbox_toexpand("hira.fu");

  push_stroke((370,740)..(530,680)..(610,630),
    (1,1)--(1.7,1.7)--(1.8,1.8));
  set_boserif(0,2,4);

  push_stroke((610,630)..tension 1.4..(410,570)..(410,470)..
      (570,200)..(470,20)..{curl 0.3}(290,270),
    (2.6,0.79)--(0.72,0.72)--(0.85,1.35)--(1.5,1.5)--(2,2)--(0,0));

  push_stroke((90+30*mincho,80)..(270,320){dir 63}..(480,410)..
      {curl 0}(910,100),
    (1.4,1.7)--(0.9,0.9)--(0.7,0.7)--(1.6,1.6));
  set_boserif(0,0,5);
  set_boserif(0,3,8);
  expand_pbox;
```

```
enddef;
```



```

vardef hira.he =
  push_pbox_toexpand(”hira.he”);

  push_stroke((90+40*mincho,290){curl 0.2}..(280,430)..tension 2..(380,540)..
      (420,530)..tension 2..(620-10*mincho,360)..{curl 0.2}(910-40*mincho,190),
    (1.7,1.7)..(1.6,1.6)..(1.2,1.2)..(1.3,1.3)..(1.7,1.7)..(2.1,2.1));
  set_boserif(0,0,5);
  expand_pbox;
enddef;
```

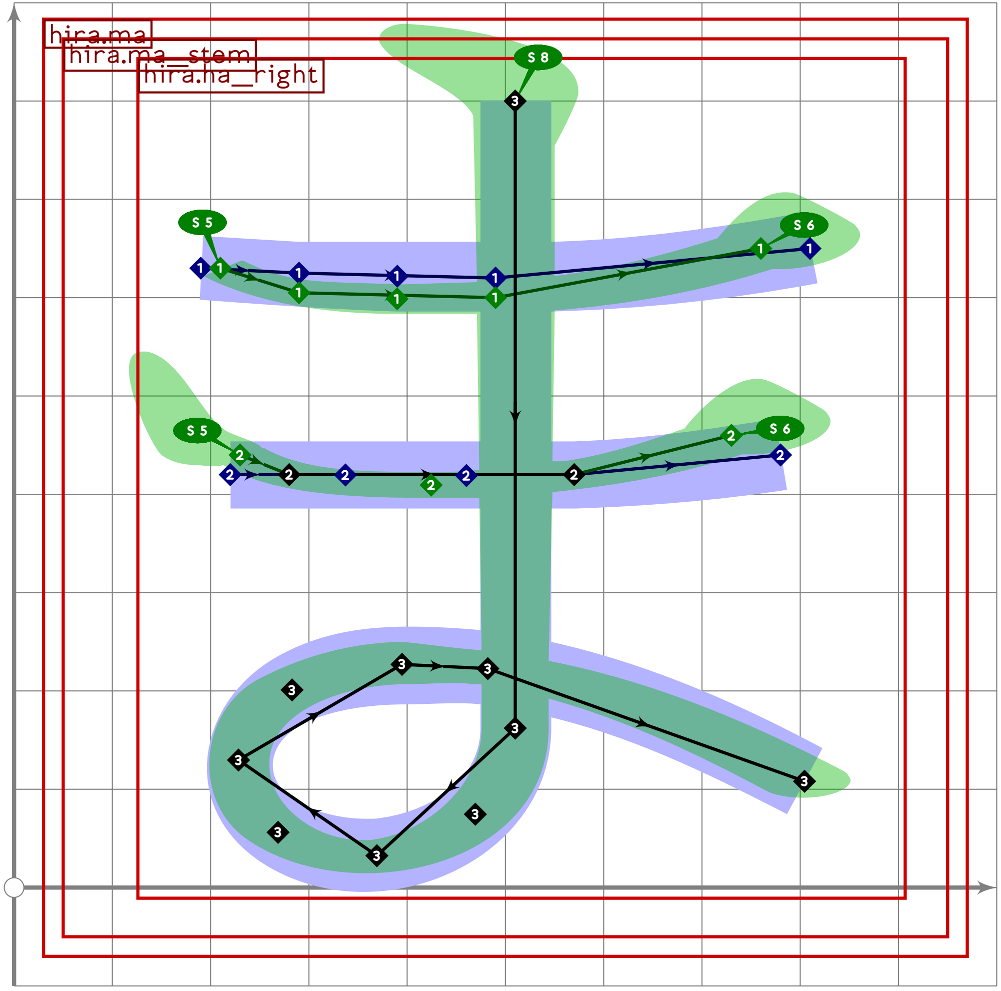
U+307B
tsuku.uni307B



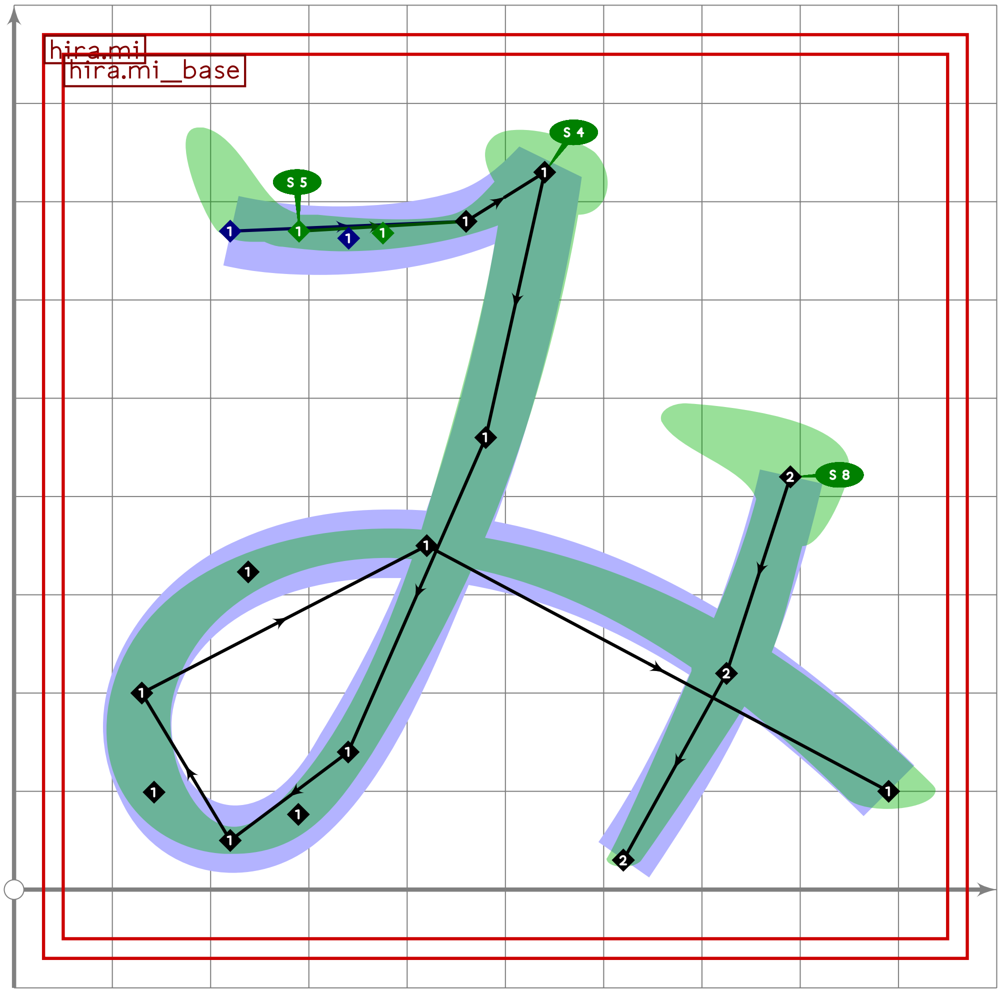
```

vardef hira.ho =
  push_pbox_toexpand("hira.ho");

  hira.ni_left;

  push_stroke((380,700)..(610-40*mincho,700)..(850-40*mincho,720),
    (1.2,1.2)..(1.6,1.6)..(1.9,1.9));
  set_boserif(0,2,6);

  push_stroke((360+20*mincho,490)..(570-30*mincho,490)..(880-40*mincho,520),
    (1.2,1.2)..(1.5,1.5)..(2,2));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  hira.ha_right;
  replace_strokep(0)(subpath
    (xpart (oldp intersectiontimes get_strokep(-2))+0.02,infinity) of oldp);
  expand_pbox;
enddef;

```

Hiragana Mamimumemo

```
%%%%%%%%%%% HIRAGANA MAMIMUMEMO

vardef hira.ma_stem =
  push_pbox_toexpand("hira.ma_stem");

  begingroup
    transform xf;

    (660,0) transformed xf = (510,0);
    (660,740) transformed xf = (510,800);
    (910,0) transformed xf = (830,0);

    tsu_xform(xf)(hira.ha_right);
    set_boserif(0,0,8);
  endgroup;
  expand_pbox;
enddef;
```

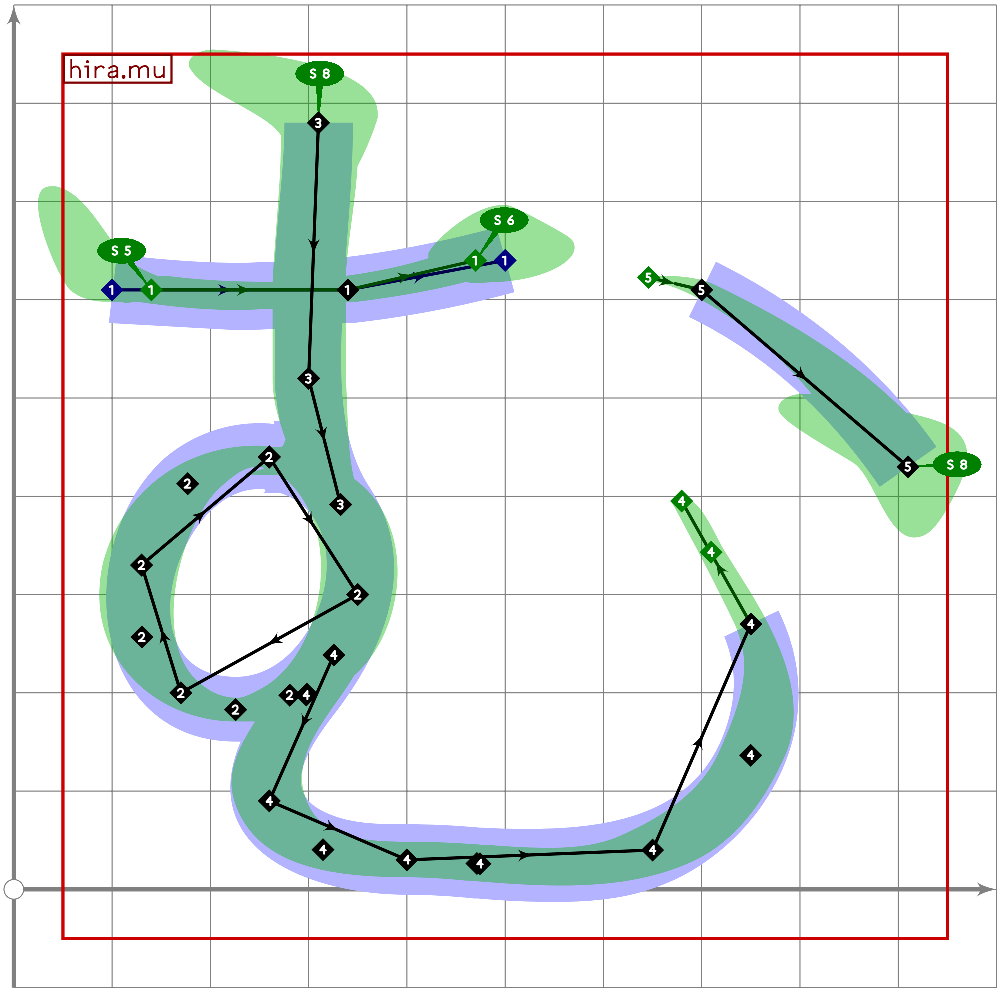


```

vardef hira.ma =
  push_pbox_toexpand(”hira.ma”);

  push_stroke((190+20*mincho,630)..(290,625-20*mincho)..tension 1.3..
      (490,620-20*mincho)..(810-50*mincho,650),
    (1.3,1.3)–(1.4,1.4)–(1.7,1.7)–(1.9,1.9));
  set_boserif(0,0,5);
  set_boserif(0,3,6);

  push_stroke((220+10*mincho,420+20*mincho)..(280,420)..tension 1.3..
      (570,420)..(780-50*mincho,440+20*mincho),
    (1.3,1.3)–(1.3,1.3)–(1.6,1.6)–(1.8,1.8));
  set_boserif(0,0,5);
  set_boserif(0,3,6);

  hira.ma_stem;
  expand_pbox;
enddef;

vardef hira.mi_base =
  push_pbox_toexpand(”hira.mi_base”);

  push_stroke(
    (220+70*mincho,670)..tension 1.3..(460,680)..
      {curl 1}(540,730){curl 1}..
      (480,460)..(340,140)..(220,50)..(130,200)..(420,350)..
      {curl 0.4}(890,100),
    (1.7,1.7)–(1.3,1.3)–(1.6,1.6)–
      (1.5,1.5)–(1.2,1.2)–(1.5,1.5)–(1.4,1.4)–(1.6,1.6)–
      (1.8,1.8));
  set_botip(0,2,0);
  set_boserif(0,0,5);
  set_boserif(0,2,4);
  expand_pbox;
enddef;
```

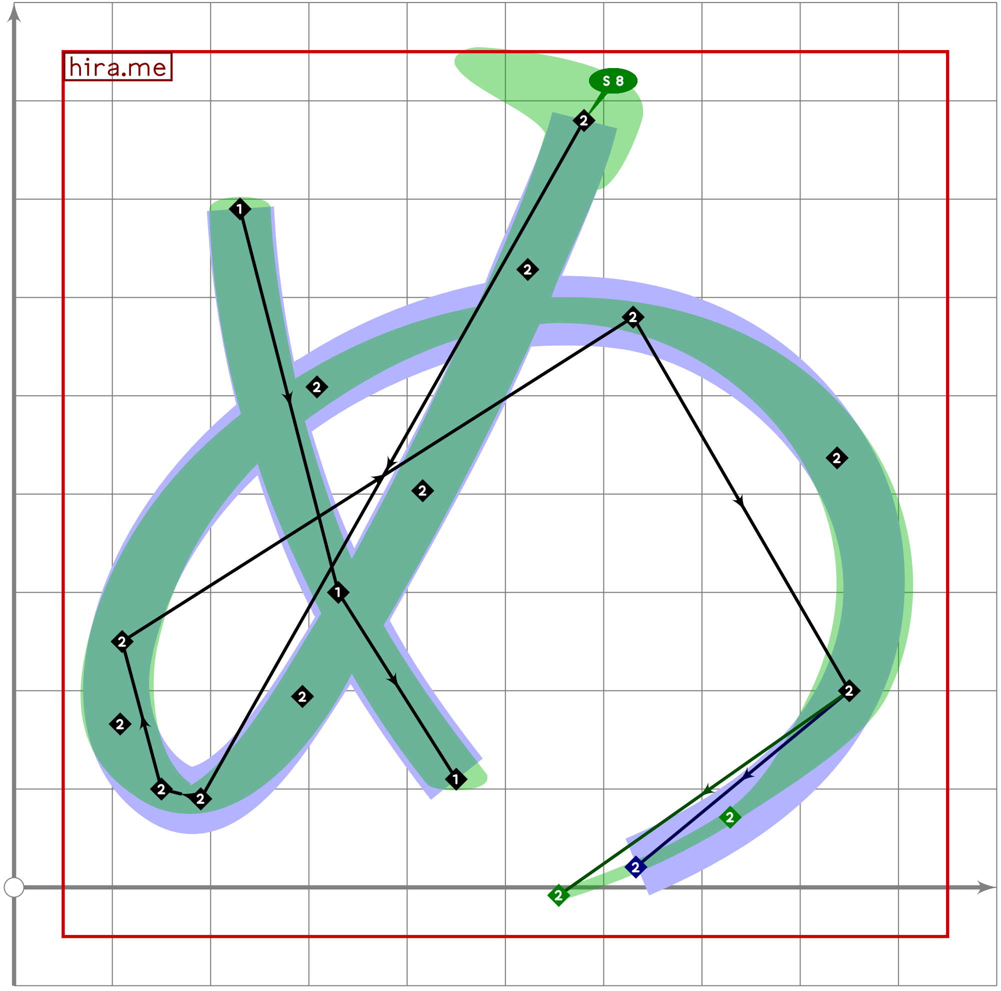


```

vardef hira.mi =
  push_pbox_toexpand(”hira.mi”);

  hira.mi_base;

  push_stroke((790,420)..(725,220)..(620,30),
    (1.3,1.3)–(1.5,1.5)–(1,1));
  set_boserif(0,0,8);
  expand_pbox;
enddef;
```

U+3080
tsuku.uni3080

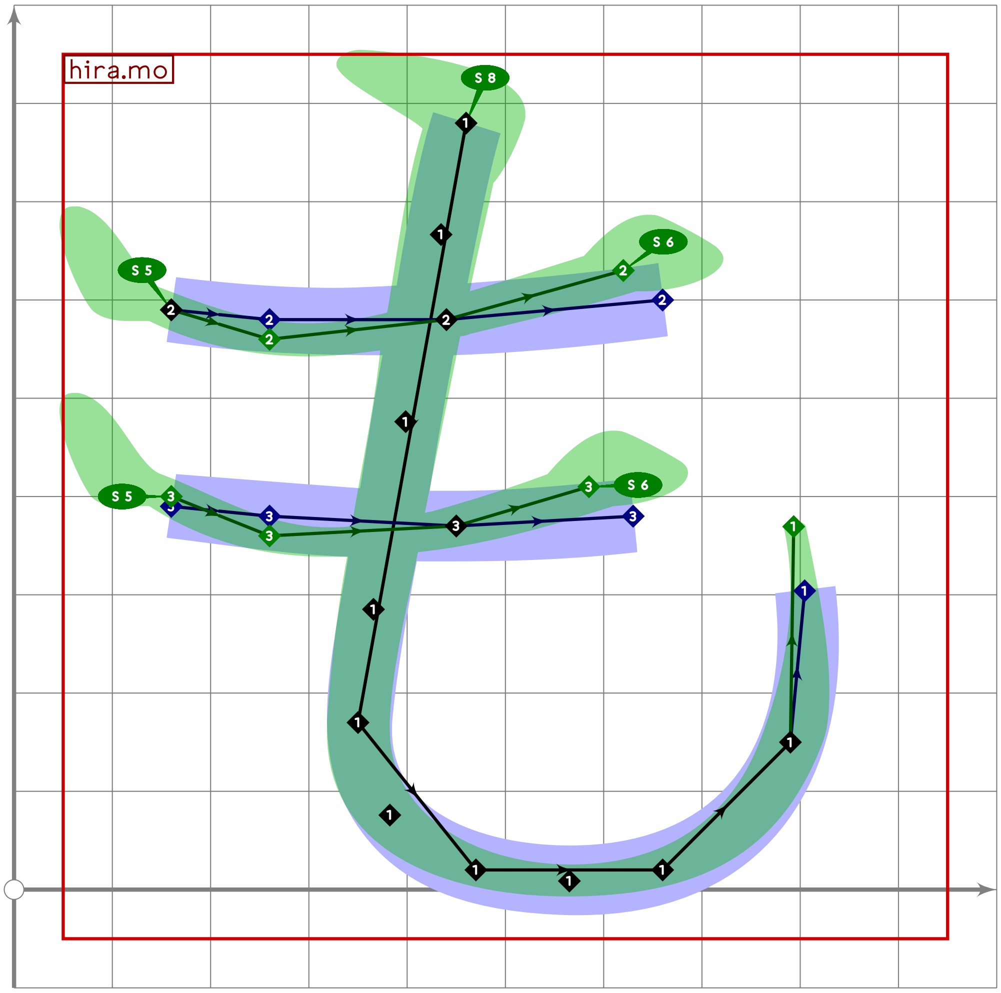


```
vardef hira.mu =
  push_pbox_toexpand("hira.mu");

  push_stroke((100+40*mincho,610)..(340,610)..(500-30*mincho,640),
    (1.6,1.6)−(1.4,1.4)−(1.5,1.5));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  push_stroke((260,440)..(350,300)..(170,200)..(130,330)..cycle,
    (1.3,1.3)..(1.6,1.6)..(1.3,1.3)..(1.6,1.6)..cycle);

  push_stroke((310,780){down}..(300,520)..
      (point 0.5 of get_strokep(0)){direction 0.5 of get_strokep(0)},
    (1.6,1.6)−(1.5,1.5)−(1.2,1.2));
  set_boserif(0,0,8);

  push_stroke(
    (point 1.25 of get_strokep(-1)){direction 1.20 of get_strokep(-1)}..
      (260,90)..(400,30){right}..(650,40)..(750,270)..
      tension 2..(600,590)..(700,610)..(910,430),
    (1.4,1.4)--(1.5,1.5)--(1.8,1.8)--(1.9,1.9)--
      (1,1)--(0.6,0.6)--(1,1)--(1.6,1.6));
  set_boserif(0,7,8);
  expand_pbox;
enddef;
```



```

vardef hira.me =
  push_pbox_toexpand("hira.me");

  push_stroke((230,690)..(330,300)..(450,110),
    (1.4,1.4)--(1.3,1.3)--(1.4,1.4));

  push_stroke((580,780){curl 0.2}..tension 2.5..(190,90)..
      (150,100)..(110,250)..tension 1.1..(630,580)..
      (850,200)..tension 1.1..{curl 0.2}(470,-30),
    (1.5,1.5)--(1.4,1.4)--(1.6,1.6)--(1.4,1.4)--
      (1.6,1.6)--(1.6,1.6)--(0.65,0.65));
  set_boserif(0,0,8);
  expand_pbox;
enddef;
```

U+3082
tsuku.uni3082

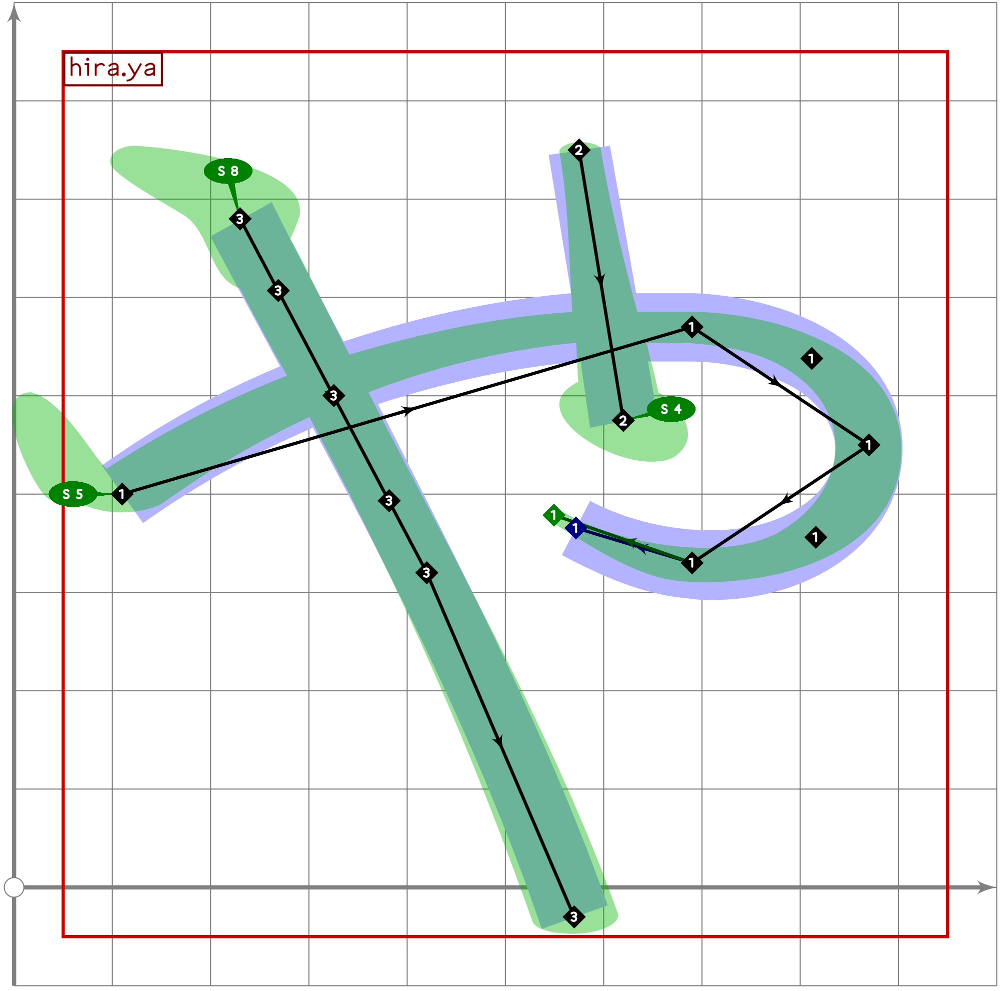


```
vardef hira.mo =
  push_pbox_toexpand("hira.mo");

  push_stroke((460,780)..tension 3..(350,170)..(470,20)..
      tension 1.2..(660,20)..(790,150)..{curl 0}(770,460),
    (1.7,1.7)--(1.4,1.4)--(1.6,1.6)--(1.7,1.7)--(1.4,1.4)--(0.6,0.6));
  set_boserif(0,0,8);

  push_stroke((160,590)..(260,580-20*mincho)..(440,580)..
      (660-40*mincho,600+30*mincho),
    (1.4,1.4)--(1.6,1.6)--(1.8,1.8)--(2,2));
  set_boserif(0,0,5);
  set_boserif(0,3,6);

  push_stroke((160,390+10*mincho)..(260,380-20*mincho)..(450,370)..
      (630-45*mincho,380+30*mincho),
    (1.4,1.4)--(1.6,1.6)--(1.8,1.8)--(2,2));
  set_boserif(0,0,5);
  set_boserif(0,3,6);
  expand_pbox;
```

```
enddef;

```

Hiragana Yayuyo

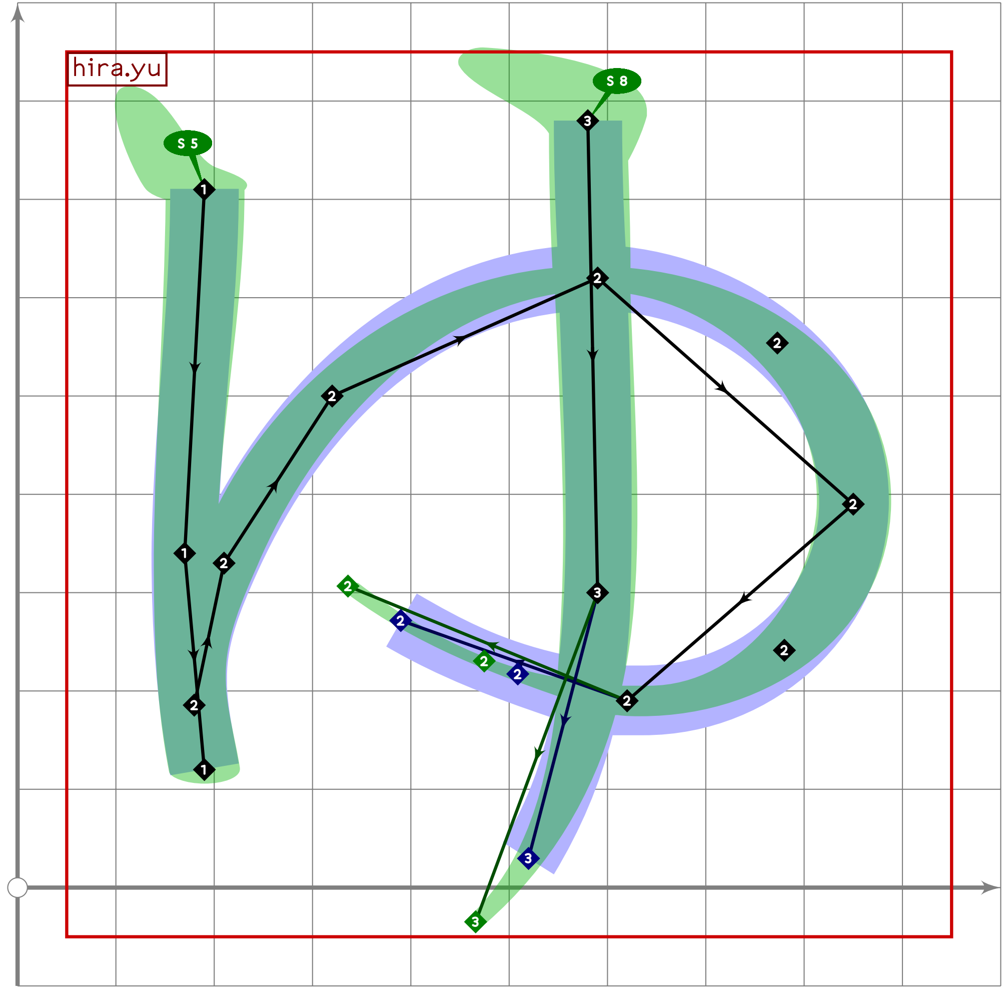
```
%%%%%%%%%% HIRAGANA YAYUYO
```



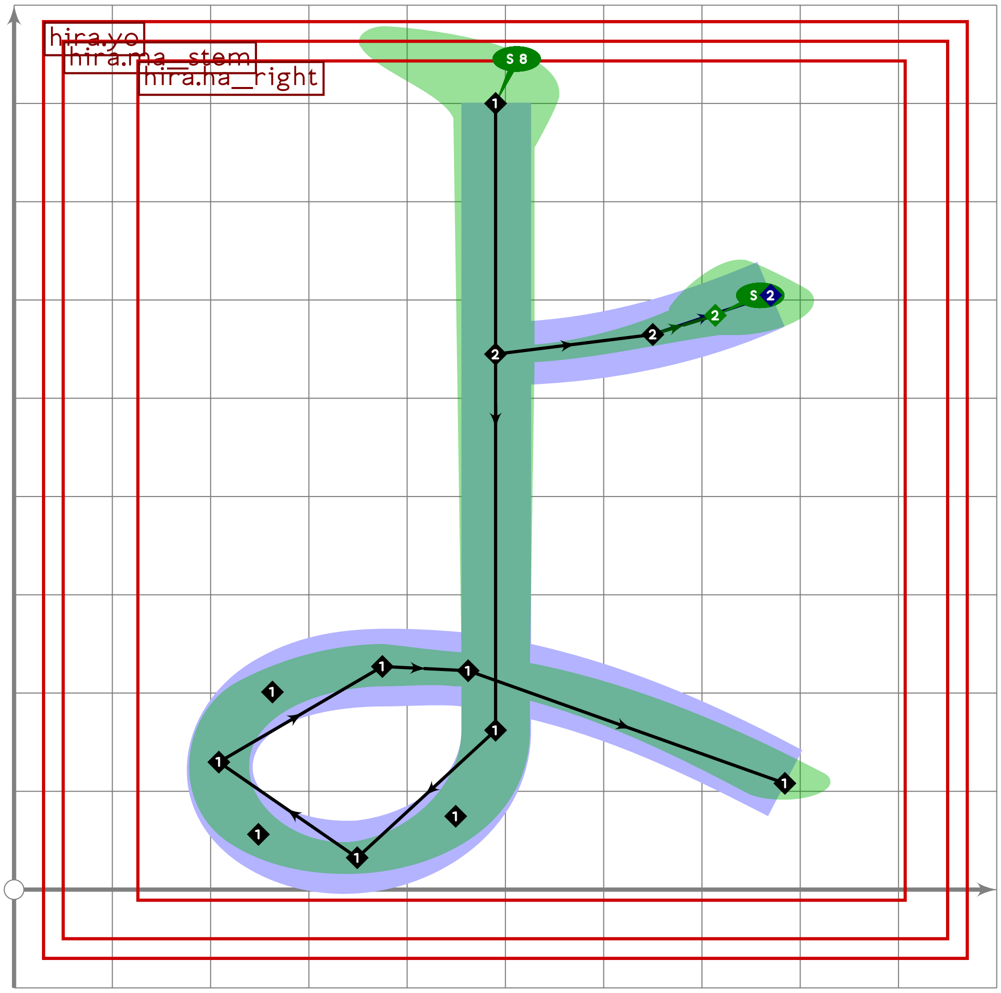
```

vardef hira.ya =
  push_pbox_toexpand("hira.ya");

  push_stroke((110,400)..(690,570)..(870,450)..(690,330)..(520,400),
    (1.8,1.8)–(1.6,1.6)–(1.4,1.4)–(1.8,1.8)–(0.6,0.6));
  set_boserif(0,0,5);

  push_stroke((575,750)–(620,475),
    (1.1,1.1)–(1.6,1.6));
  set_boserif(0,1,4);

  push_stroke((230,680)..tension 2..(420,320)..(570,-30),
    (1.6,1.6)–(1.4,1.4)–(1.7,1.7));
  set_boserif(0,0,8);
  expand_pbox;
```

U+3086
tsuku.uni3086

```
enddef;
```



```

vardef hira.yu =
  push_pbox_toexpand("hira.yu");

  push_stroke((190,710){down}...(170,340)..(190,120),
    (1.6,1.6)--(1.4,1.4)--(1.5,1.5));
  set_boserif(0,0,5);

  push_stroke((point 1.7 of get_strokep(0))
        {-direction 1.7 of get_strokep(0)}..(210,330)..
      (320,500)..(590,620)..(850,390)..(620,190)..tension 1.3..(320,320),
    (1.4,1.4)--(1.5,1.5)--(1.4,1.4)--(1.5,1.5)--(1.6,1.6)--
      (1.6,1.6)--(0.77,0.77));

  push_stroke((580,780){down}...(590,300)..{dir 190}(360,-85),
    (1.6,1.6)--(1.5,1.5)--(0.6,0.6));
  set_boserif(0,0,8);
  expand_pbox;
enddef;
```

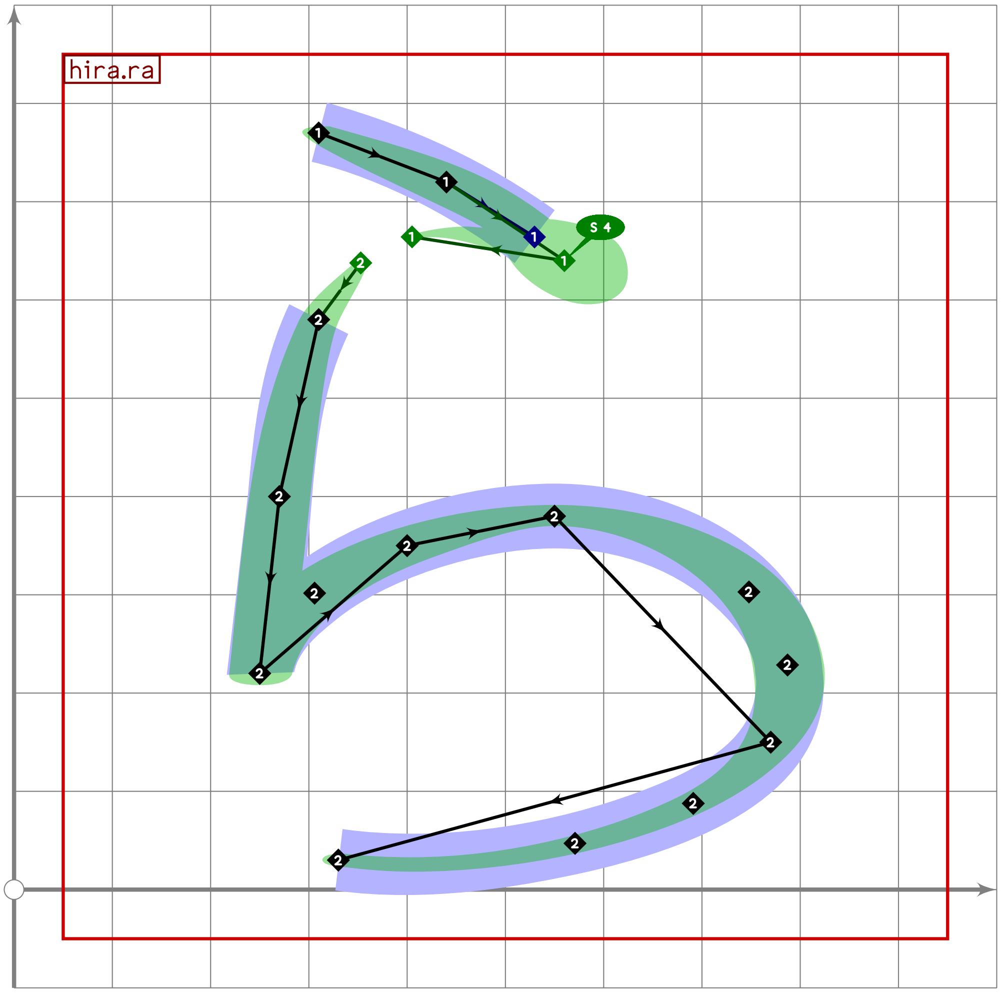


```

vardef hira.yo =
  push_pbox_toexpand(”hira.yo”);

  hira.ma_stem;

  replace_strokep(0)(oldp shifted (−20,0));
  z0=point 0.4 of get_strokep(0);

  push_stroke(z0..(z0+(160,20))..(z0+(280,60)+60*mincho*dir 200),
    (1.2,1.2)−(1.4,1.4)−(1.8,1.8));
  set_boserif(0,2,6);
  expand_pbox;
enddef;

```

Hiragana Rarirurero

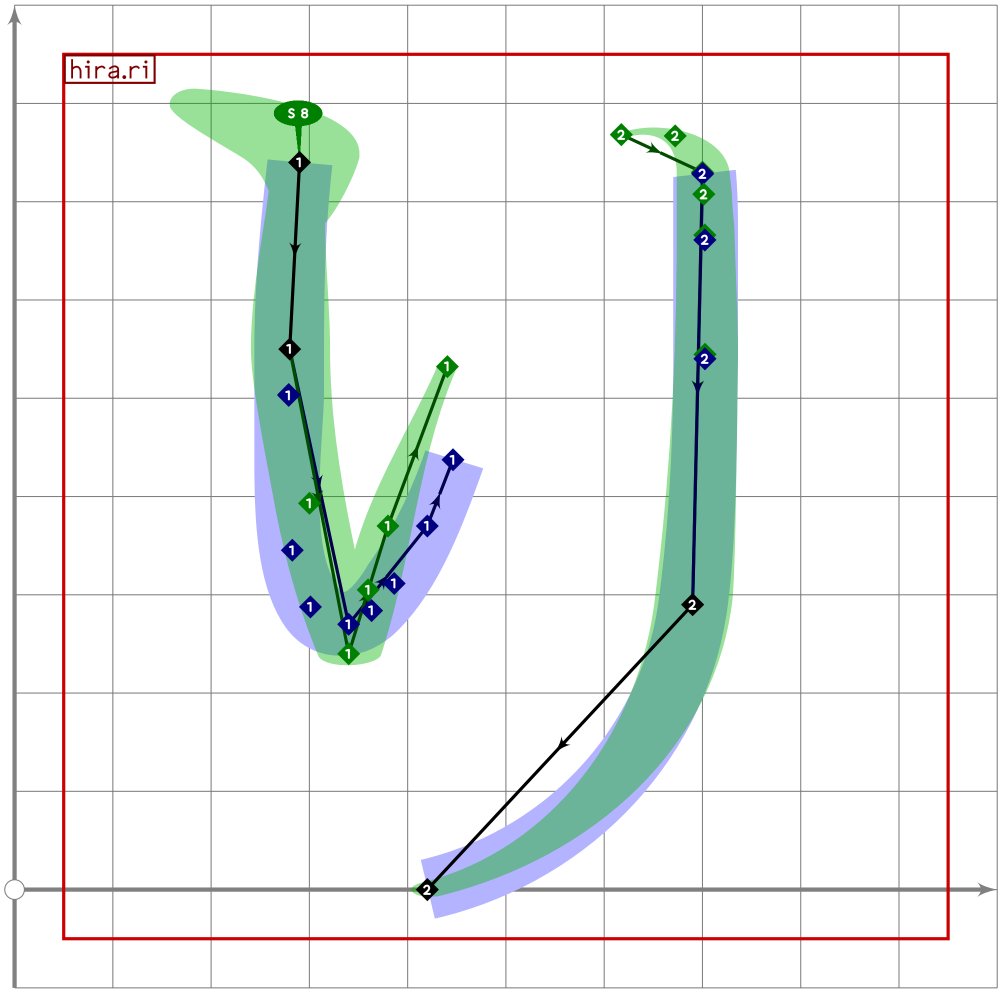
```
%%%%%%%%%% HIRAGANA RARIRURERO
```

U+3089
tsuku.uni3089



```
vardef hira.ra =
  push_pbox_toexpand("hira.ra");

  push_stroke((370,770)..(500,720)..{curl 1}(620,640){curl 0.1}..
      (430,650)..(370,580)..(330,400)..(310,220),
    (1,1)--(1.6,1.6)--(2,0.78)--(0.55,0.55)--
      (1.8,1)--(1.4,1.4)--(1.6,1.6));
  set_boserif(0,2,4);

  hira.chi_bottom;
  set_botip(0,2,0);

  replace_strokep(0)(oldp shifted (-60,0));
  expand_pbox;
enddef;
```

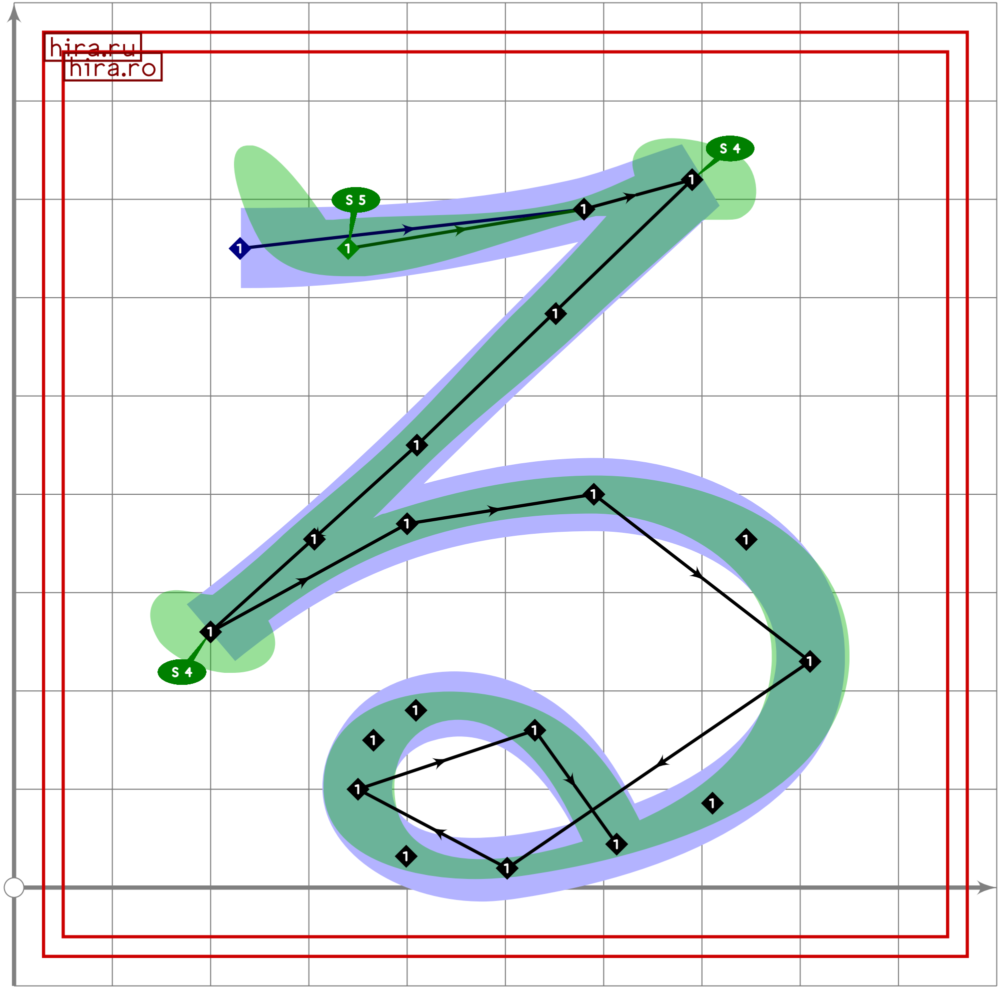
U+308A
tsuku.uni308A



```
vardef hira.ri =
  push_pbox_toexpand(”hira.ri”);

  begingroup
    save ripx,ripy,ripz,x,y;
    path ripx,ripy,ripz;
    numeric x[],y[];
    z1=(290,740);
    z2=(280,550);
    z3=(340,270-30*mincho);
    z4=(420-40*mincho,370);
    z5=(540,710);
    z6=(700,730);
    ripx=z1..z2{down}..tension 1.5..z3..
      tension 1.5..z4..z5..z6..tension 5 and 1.2..
      (690,290)..tension 0.75 and 1..{curl 0.45}(420,0);
    ripy=z1..z2{down}..tension 1.5..{curl 1}z3{curl 1}..
      tension 1.5..z4..z5..z6..tension 5 and 1.2..
      (690,290)..tension 0.75 and 1..{curl 0.45}(420,0);
    push_stroke(interpath(mincho,ripx,ripy),
```

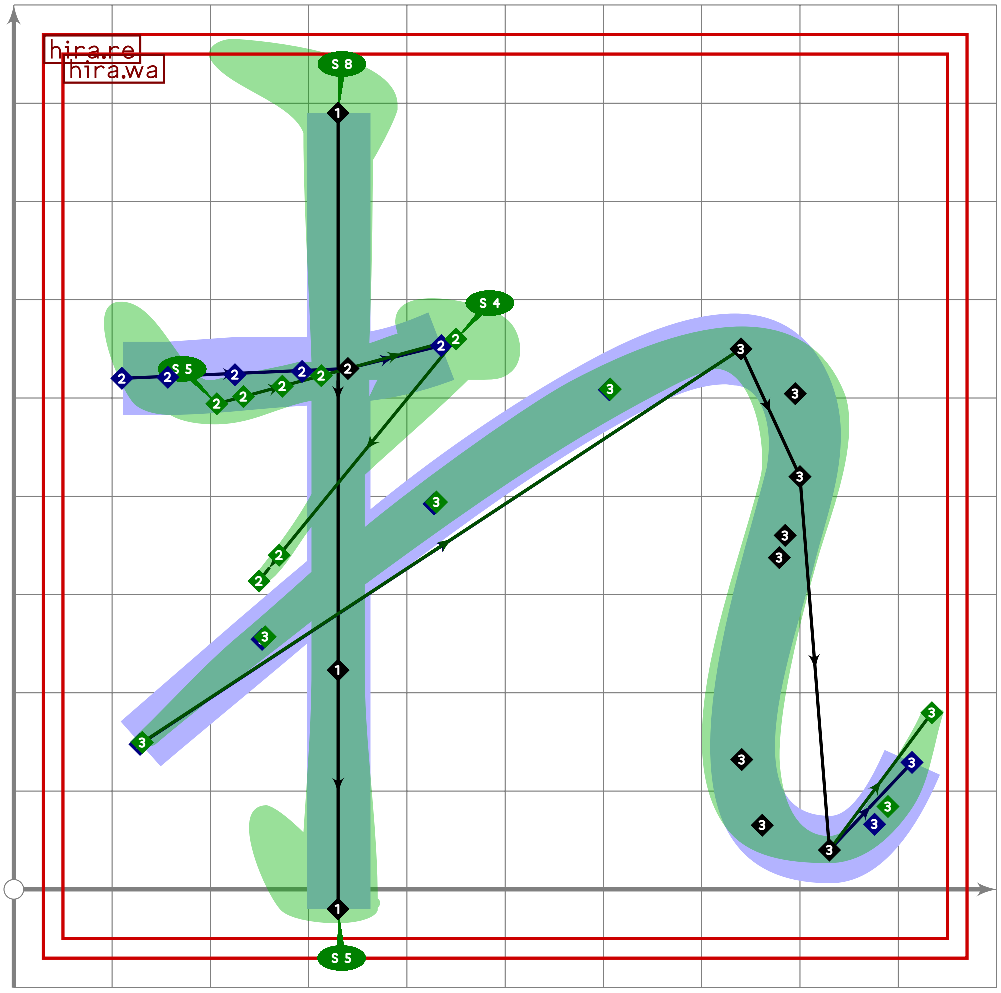
U+308B
tsuku.uni308B

```
      (1.3,1.3)–(1.6,1.6)–(1.4,1.4)–(1.2,1.2)–(0.4,0.2)–
      (1.5,0.99)–(1.6,1.6)–(1,1));
    set_boserif(0,0,8);
  endgroup;
  expand_pbox;
enddef;
```



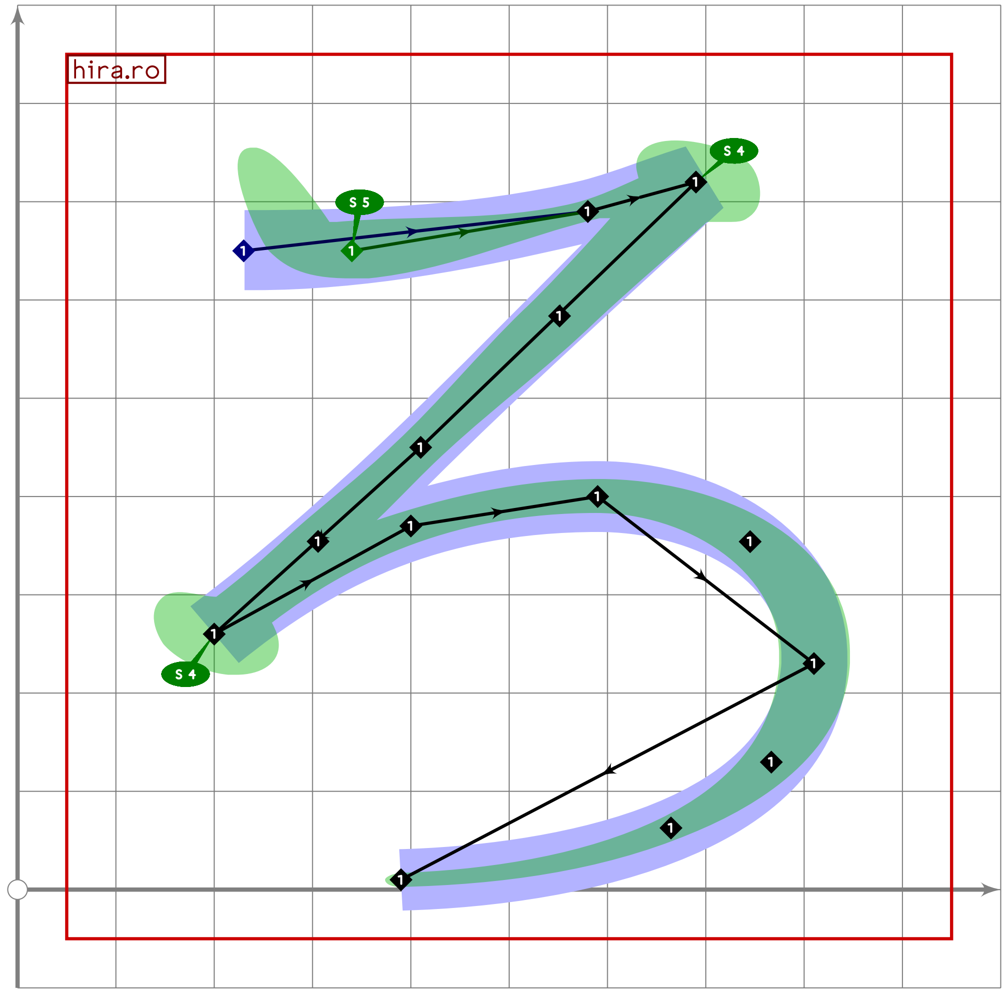
```

vardef hira.ru =
  push_pbox_toexpand("hira.ru");

  hira.ro;

  replace_strokep(0)((subpath (0,7.8) of oldp)..(350,100)..(530,160)..
    {curl 0.2}(point 7.6 of oldp));
  replace_strokeq(0)((2.6,2.6)–(1.2,1.2)–(1.9,1.9)–
    (1.3,1.3)–(1.6,1.6)–
    (1.5,1.5)–(1.9,1.9)–(1.6,1.6)–
    (1.2,1.2)–(1.5,1.5)–(1.4,1.4));
  expand_pbox;
enddef;
```

U+308C
tsuku.uni308C



```
vardef hira.re =
  push_pbox_toexpand(”hira.re”);

  hira.wa;

  replace_strokep(0)((subpath (0,4) of oldp){curl 0}..
    tension 2..(740,550)..(800,420)..
    (830,40){right}..tension 1.5..{curl 0}(960,270));
  replace_strokeq(0)((2,2)--(1.6,1.6)--(2.7,0.9)--
    (0.84,0.7)--(0.79,0.97)--
    (2.1,2.1)--(1.6,1.6)--(1.5,1.5)--(0.5,0.5));
  set_boserif(0,2,4);
  expand_pbox;
enddef;
```

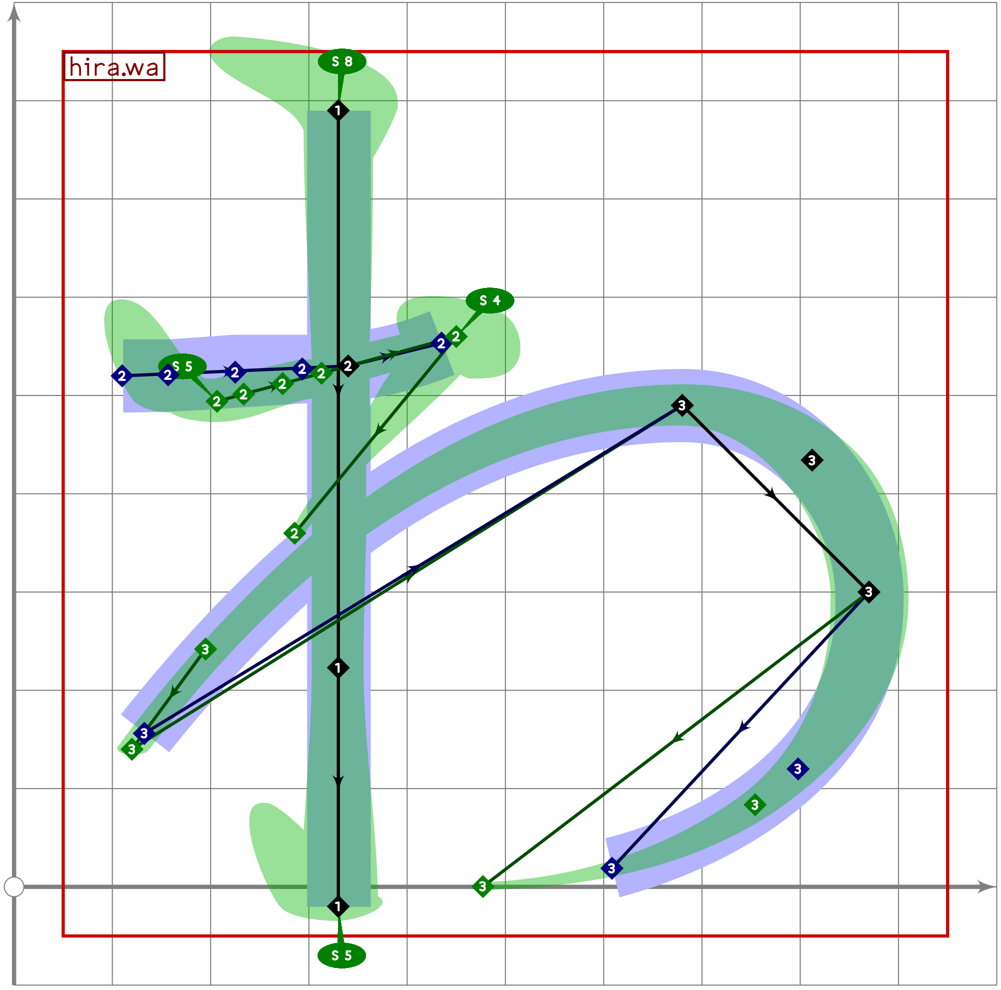
U+308D
tsuku.uni308D



```
vardef hira.ro =
  push_pbox_toexpand("hira.ro");

  push_stroke((230+110*mincho,650)..(580,690)..{curl 1}(690,720){curl 1}..
      (410,450)..{curl 1}(200,260){curl 1}..
      (400,370)..(590,400){right}..(810,230)..
      tension 1.1..{curl 0}(390,10),
    (2.6,2.6)--(1.2,1.2)--(1.9,1.9)--
      (1.3,1.3)--(1.6,1.6)--
      (1.5,1.5)--(1.7,1.7)--(1.5,1.5)--(1,1));
  set_botip(0,2,0);
  set_botip(0,4,0);
  set_boserif(0,0,5);
  set_boserif(0,2,4);
  set_boserif(0,4,4);
  expand_pbox;
enddef;
```

Hiragana Wawiwewo/N/Iteration

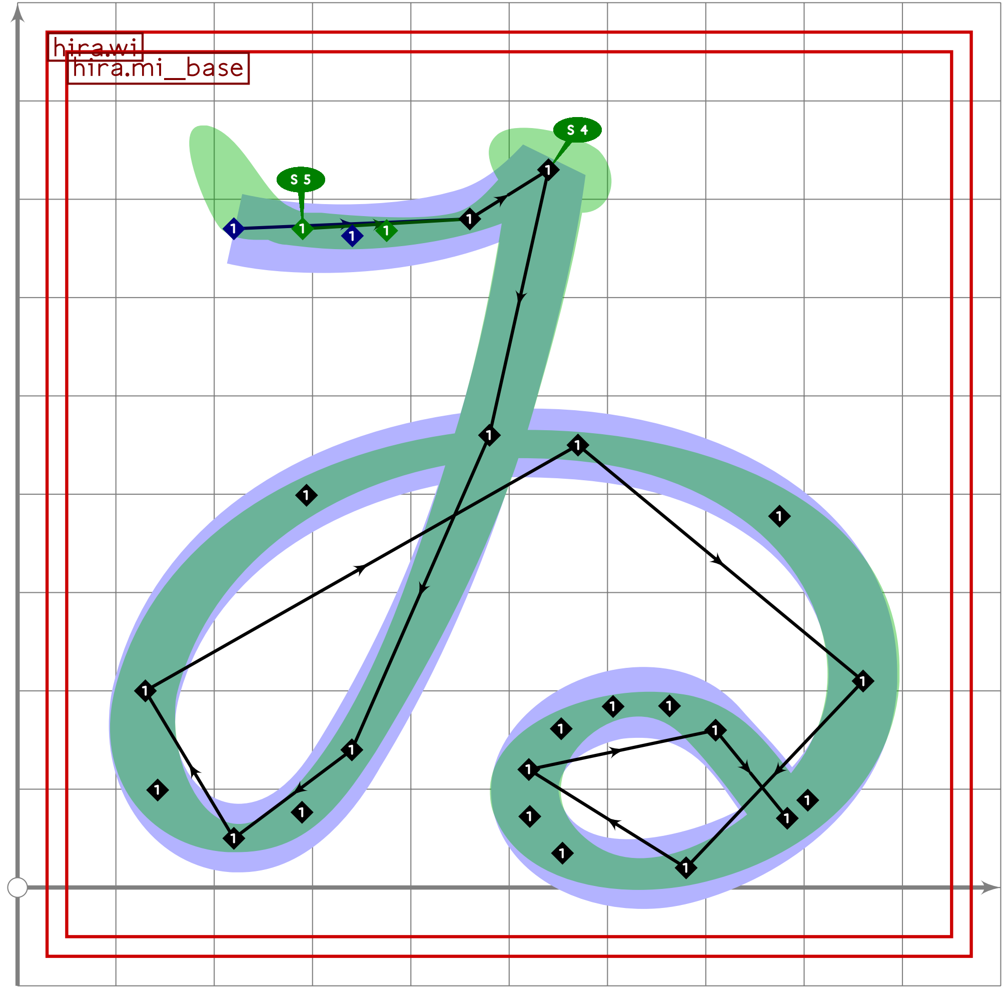
```
%%%%%%%%%% HIRAGANA WAWIWEWO/N/ITERATION
```



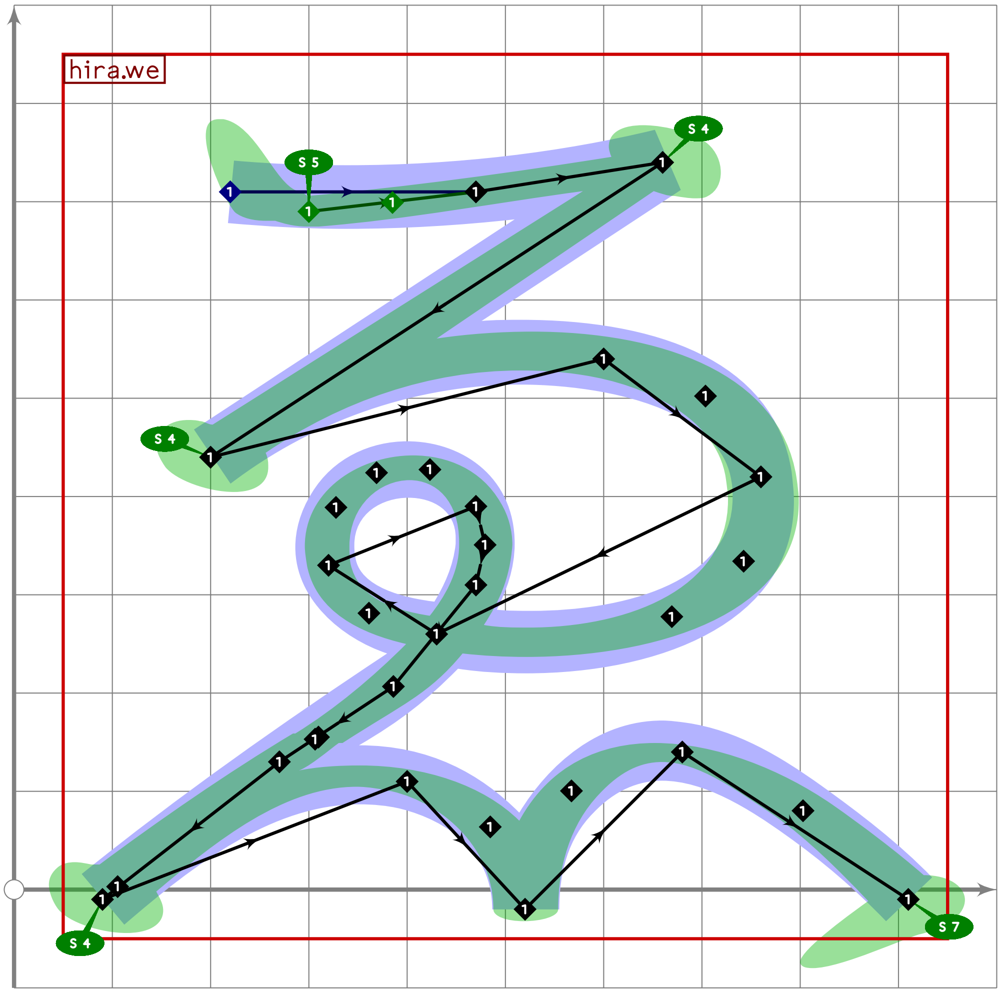
```

vardef hira.wa =
  push_pbox_toexpand("hira.wa");

  push_stroke((330,790)--(0.7[(330,790),(330,-20)])--(330,-20),
    (1.5,1.5)--(1.2,1.2)--(1.6,1.6));
  set_boserif(0,0,8);
  set_boserif(0,2,5);

  push_stroke(((110,520)+100*mincho*dir -15)..tension 2..(340,530)..
      {curl 1}(450,560){curl 1}..
      (270,340)..{curl 1}(120,140){curl 0.2}..
      (680,490){right}..(870,300)..{curl 0.2}(450,0),
    (2,2)--(1.6,1.6)--(2.2,0.9)--
      (0.7,0.7)--(0.97,0.97)--
      (2,2)--(1.6,1.6)--(0.8,0.8));
  set_botip(0,2,0);
  set_botip(0,4,0);
  set_boserif(0,0,5);
```

U+3090
tsuku.uni3090

```
  set_boserif(0,2,4);
  expand_pbox;
enddef;
```



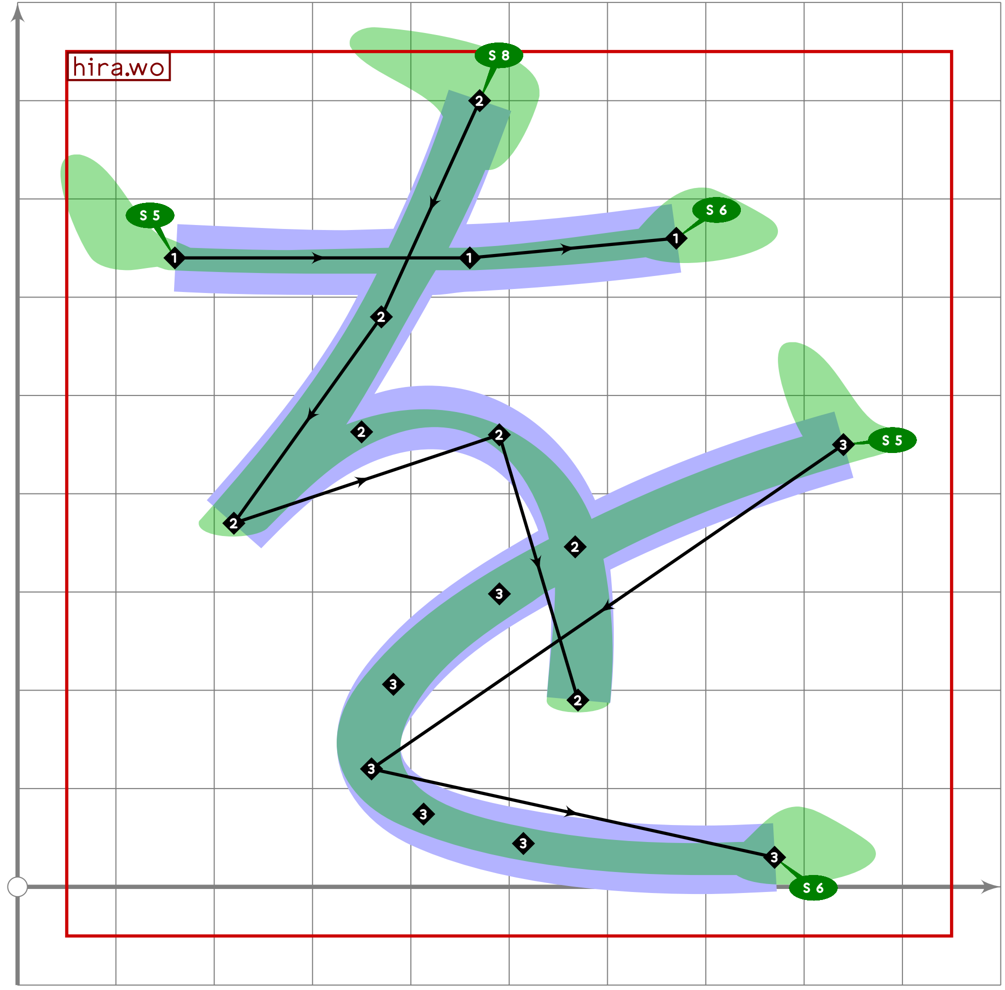
```

vardef hira.wi =
  push_pbox_toexpand(”hira.wi”);

  hira.mi_base;

  replace_strokep(0)((subpath (0,6) of oldp)..(570,450)..(860,210)..(680,20)..
    (520,120)..tension 1.2..(710,160));
  replace_strokep(0)(oldp..{curl 0.3}(point 8.6 of oldp));

  replace_strokeq(0)((1.7,1.7)–(1.3,1.3)–(1.6,1.6)–
    (1.5,1.5)–(1.2,1.2)–(1.5,1.5)–(1.4,1.4)–
    (1.6,1.6)–(1.5,1.5)–(1.7,1.7)–(1.4,1.4)–(1.5,1.5));
  expand_pbox;
enddef;
```



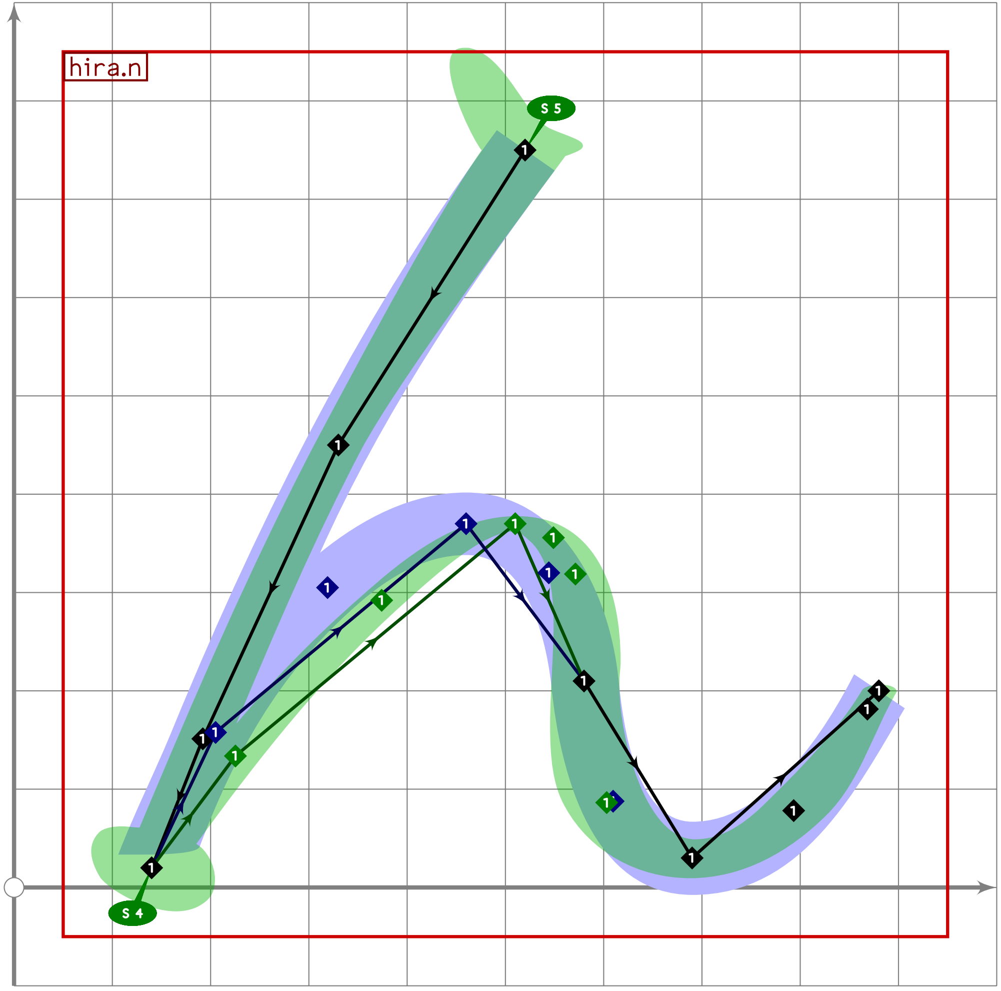
```
vardef hira.we =
  push_pbox_toexpand("hira.we");

  push_stroke((220+80*mincho,710-20*mincho)..(470,710)..{curl 1}(660,740)--
      (200,440){curl 1}..
      (600,540)..(760,420)..(430,260)..(320,330)..(470,390)..(470,310)..
      (270,130)..{curl 0}(90,-10){curl 0.1}..
      (400,110)..{down}(520,-20){up}..
      (680,140)..tension 1.3..{curl 0.2}(910,-10),
    (1.8,1.8)--(1.6,1.6)--(1.5,1.5)--(1.9,1.9)--
      (2,2)--(1.6,1.6)--(1.7,1.7)--(1.2,1.2)--(1.3,1.3)--(1.35,1.35)--
         (1.4,1.4)--(1.5,1.5)--(1.3,1.3)--(1.8,1.8)--
      (1.4,1.4)--(1.5,1.5)--
      (1.3,1.3)--(1.7,1.7));
  replace_strokep(0)(insert_nodes(oldp)(8.5,9.5));
  set_bosize(0,90);
  set_botip(0,2,0);
  set_botip(0,3,0);
  set_botip(0,13,0);
  set_botip(0,15,0);
```

U+3092
tsuku.uni3092

```
  set_boserif(0,0,5);
  set_boserif(0,2,4);
  set_boserif(0,3,4);
  set_boserif(0,13,4);
  set_boserif(0,17,7);
  expand_pbox;
enddef;
```



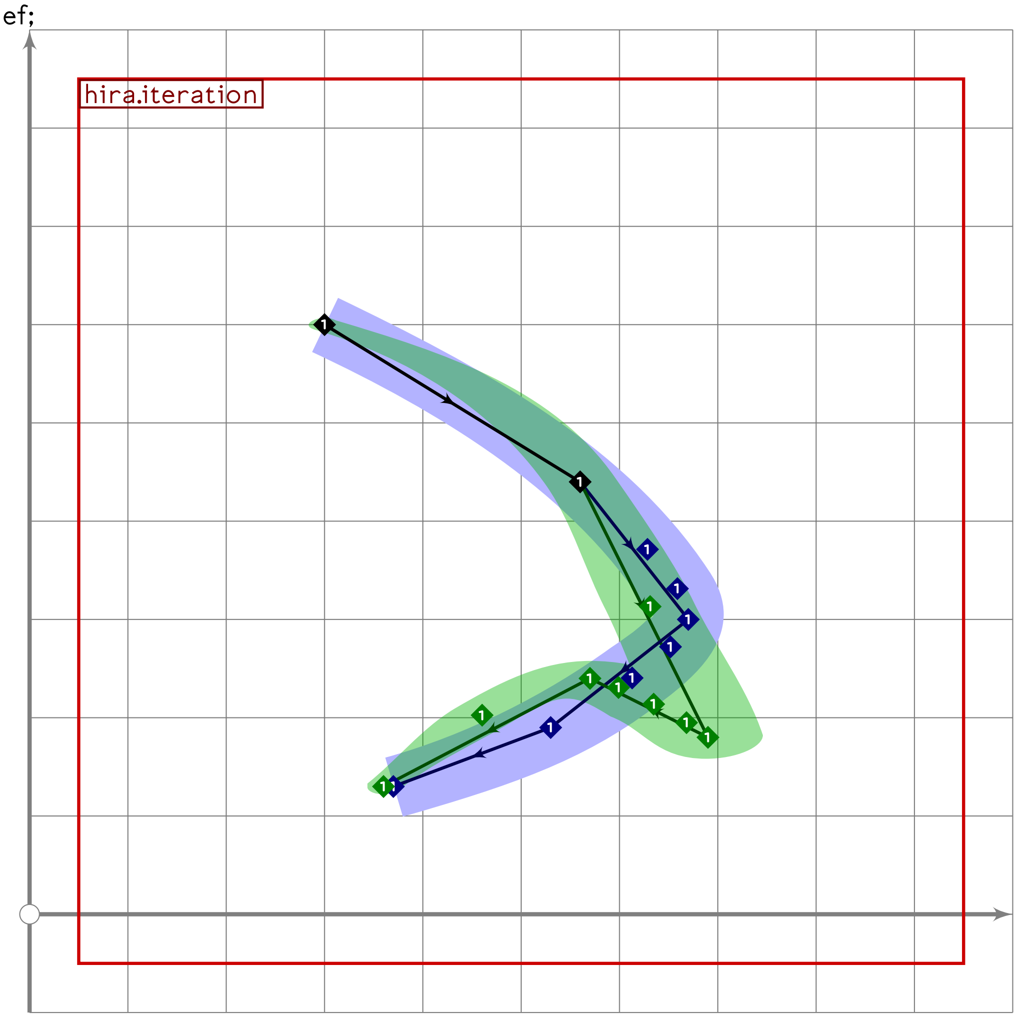
```

vardef  hira.wo =
  push_pbox_toexpand("hira.wo");

  push_stroke((160,640)..(460,640)..(670,660),
    (1.3,1.3)--(1.4,1.4)--(1.6,1.6));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  push_stroke((470,800)..(370,580)..{curl 1}(220,370){curl 0.1}..
    (490,460)..{curl 0}(570,190),
    (1.4,1.4)--(1.3,1.3)--(1.5,1.5)--(1.01,1.01)--(1.4,1.4));
  set_botip(0,2,0);
  set_boserif(0,0,8);

  push_stroke((840,450){curl 0.017}..tension 1 and 2..(360,120)..
    tension 2 and 1..{curl 0.03}(770,30),
    (1.7,1.7)--(1.4,1.4)--(1.7,1.7));
  set_boserif(0,0,5);
  set_boserif(0,2,6);
  expand_pbox;
enddef;
```



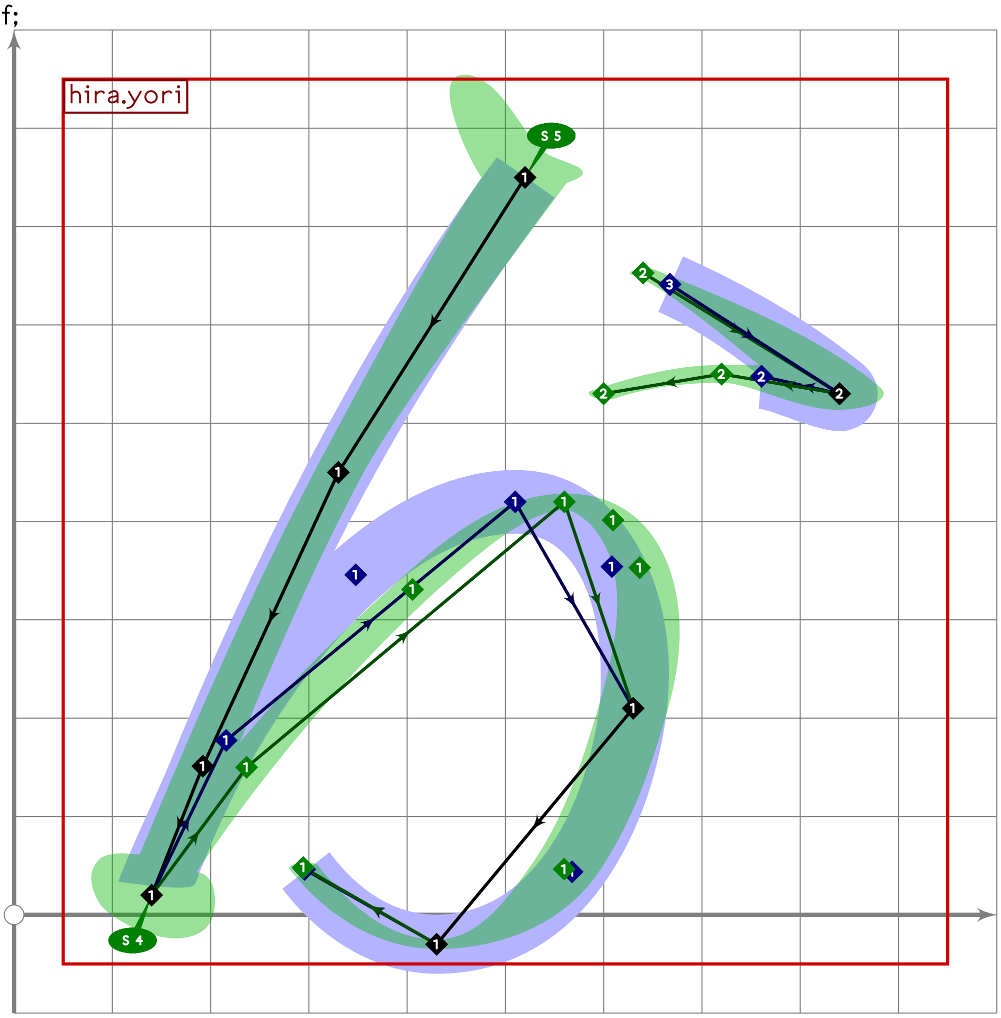
```

vardef hira.n =
  push_pbox_toexpand("hira.n");

  push_stroke((520,750)..(330,450)..{curl 0.2}(140,20){curl 0.1}..
    tension (1.2+0.6*mincho)..(460+50*mincho,370){right}..
       (580,210)..(690,30){right},
   (1.7,1.7)--(1.2,1.2)--(1.3,1.3)--
     (1.1,1.1)--(1.5,1.5)--(1.9,1.9)--(1,1));
  replace_strokep(0)(oldp{right}..(880,200){-direction 0.5 of oldp});
  replace_strokep(0)(insert_nodes(oldp)(1.7,2.3));
  replace_strokeq(0)(insert_nodes(oldq)(1.7,2.3));
  set_botip(0,3,0);
```

U+309D
tsuku.uni309D

```
  set_boserif(0,0,5);
  set_boserif(0,3,4);
  expand_pbox;
enddef;
```



```

vardef hira.iteration =
  push_pbox_toexpand("hira.iteration");

  push_stroke(begingroup
    save ripx,ripy;
    path ripx,ripy;
    ripx:=(300,600){curl 0.2}..(560,440)..
      tension 1.5 and 2..(670,300)..
      tension 2 and 1.5..(530,190)..{curl 0.2}(370,130);
    ripy:=(300,600){curl 0.2}..(560,440)..
      tension 1.5..{curl 1}(690,180){curl 1}..
      tension 2 and 1.5..(570,240)..tension 1.4..{curl 0}(360,130);
    interpath(mincho,ripx,ripy)
  endgroup,
    (1,1)--(1.5,1.5)--(2,2)--(1.9,1.9)--(1,1));
  set_botip(0,2,0);
  expand_pbox;
enddef;
```



```

vardef hira.yori =
  push_pbox_toexpand(”hira.yori”);

  push_stroke((520,750)..(330,450)..{curl 0.2}(140,20){curl 0.1}..
    tension (1.2+0.6*mincho)..(510+50*mincho,420){right}..
       (630,210)..(430,-30){left}..tension 1.3..(290,600)..
       {curl 0.1}(840,530){curl 1}..(720,550)..(600,530),
    (1.7,1.7)–(1.2,1.2)–(1.3,1.3)–
     (1.1,1.1)–
     (1.5,1.5)–(1.5,1.5)–(-1,-0.4)–
     (1.8,1.6)–(1.6,0.7)–(1.9,0));
  replace_strokep(0)(insert_nodes(oldp)(1.7,2.3));
  replace_strokeq(0)(insert_nodes(oldq)(1.7,2.3));
  set_botip(0,3,0);
  set_botip(0,9,0);
  set_boserif(0,0,5);
  set_boserif(0,3,4);
  expand_pbox;
```

```
enddef;
```

iching.mp

```
%
% I Ching characters for Tsukurimashou
% Copyright (C) 2011 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(iching);

```

```

iching.size:=680;

vardef make_iching_xform(expr numlines) =
  transform iching_xform;
  numeric x[];

  x2-x1=iching.size;
  (x1+x2)/2=500;

  (-1,(numlines+1)/2) transformed iching_xform=
    (x1,0.5[latin_wide_low_h,latin_wide_high_h]);
  (1,(numlines+1)/2) transformed iching_xform=
    (x2,0.5[latin_wide_low_h,latin_wide_high_h]);
  (-1,(numlines+1)/2-2.5) transformed iching_xform=
    (x1,latin_wide_low_h);
enddef;

vardef iching.line(expr line,numlines,linetype) =
  make_iching_xform(numlines);
  if linetype=0:
    push_stroke(((-1,line)--(-0.28,line)) transformed iching_xform,
      (2,2)--(2,2));
    push_stroke(get_strokep(0) reflectedabout (centre_pt,centre_pt+down),
      (2,2)--(2,2));
  else:
    push_stroke(((-1,line)--(1,line)) transformed iching_xform,
      (2,2)--(2,2));
  fi;
  push_anchor(anc_iching_line(line),
    identity shifted (((1,line) transformed iching_xform)
                     +(1000-iching.size)*0.25*right));
enddef;

% WARNING, nonstandard calling convention, simply returns a path to fill
vardef iching.dot(expr line,numlines) =
  begingroup;
```

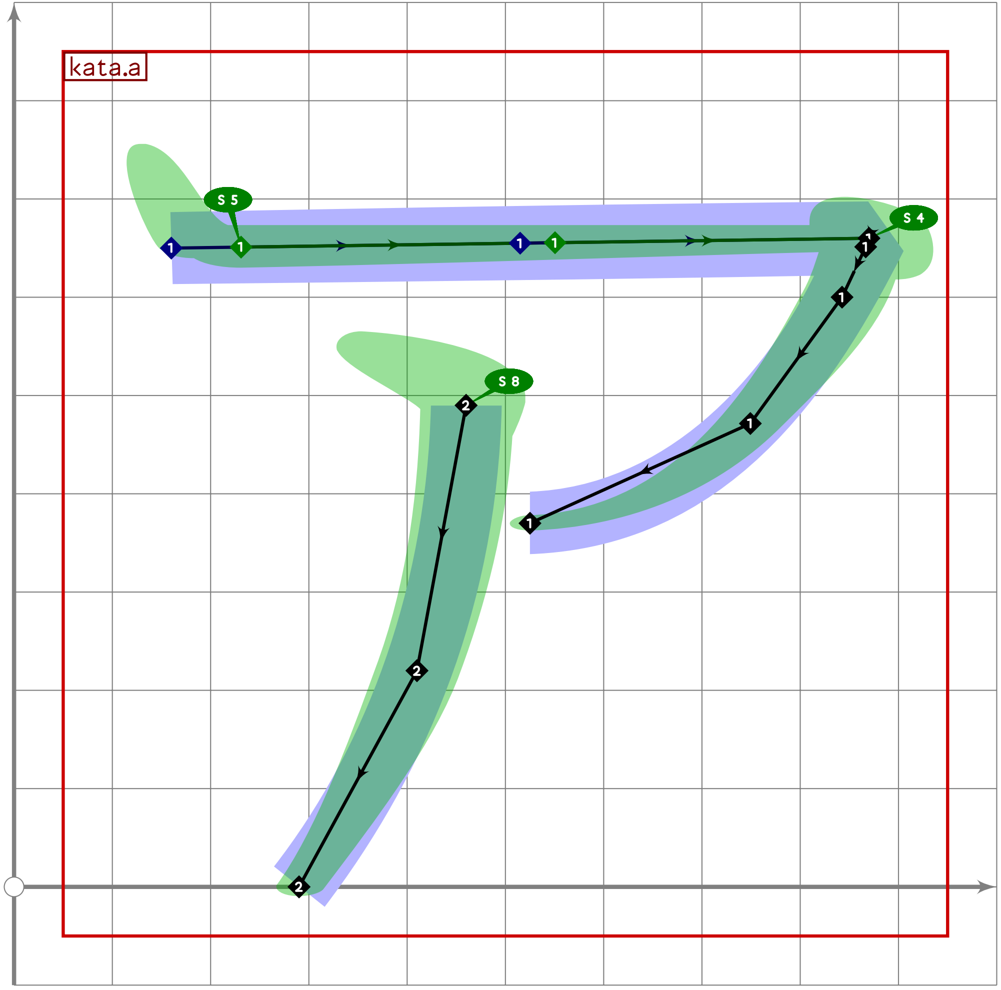
```
    make_iching_xform(numlines);
    (fullcircle scaled (tsu_punct_size*1.10)
      shifted (((1,line) transformed iching_xform)
              +(1000-iching.size)*0.25*right))
  endgroup
enddef;
```

katakana.mp

```
%
% Katakana for Tsukurimashou
% Copyright (C) 2011, 2012, 2013 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(katakana);

```

Katakana Vowels

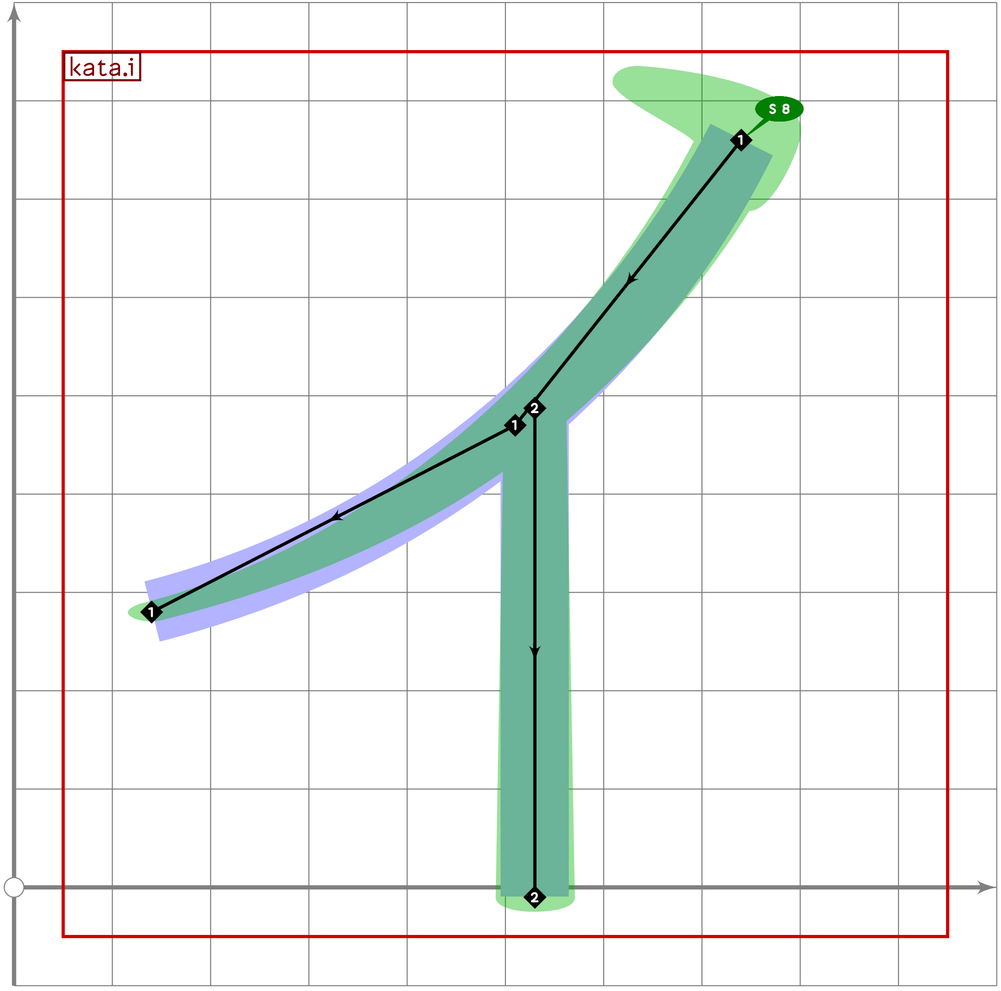
```
%%%%%%%%%% KATAKANA VOWELS
```



```

vardef kata.a =
  push_pbox_toexpand("kata.a");

  kata.fu_stroke((160,650),(870,660),(525,370));

```

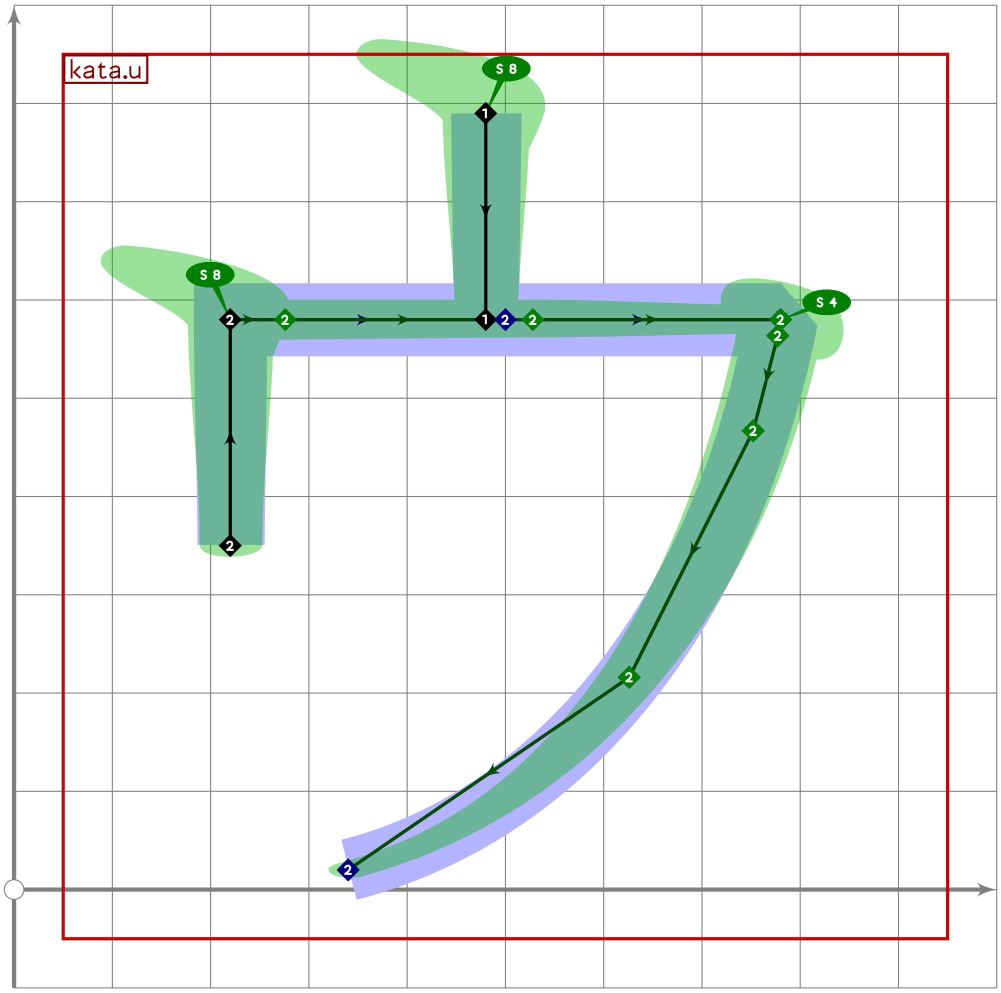
U+30A4
tsuku.uni30A4

```
  push_stroke((460,490)..(410,220)..(290,0),
    (1.8,1.8)−(1.7,1.7)−(1.2,1.2));
  set_boserif(0,0,8);
  expand_pbox;
enddef;
```



```

vardef  kata.i =
  push_pbox_toexpand("kata.i");

  push_stroke((740,760)..(510,470)..(140,280),
    (1.8,1.8)−(1.7,1.7)−(1.2,1.2));
  set_boserif(0,0,8);

  push_stroke((get_strokep(0) intersectionpoint
      ((530,-infinity)−(530,infinity)))−(530,-10),
    (1.4,1.4)−(1.6,1.6));
  expand_pbox;
enddef;
```

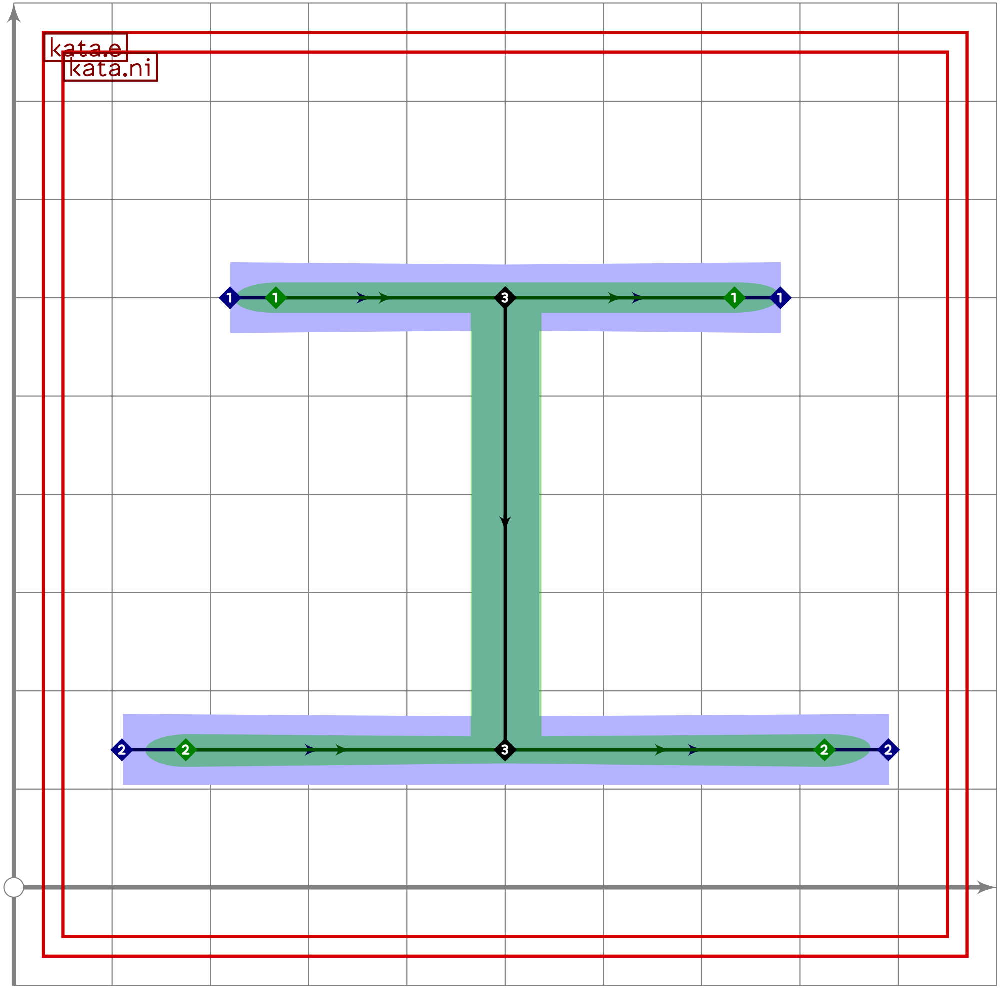
U+30A6
tsuku.uni30A6



```
vardef kata.u =
  push_pbox_toexpand(”kata.u”);

  push_stroke((480,790)–(480,580),
    (1.7,1.7)–(1.4,1.4));
  set_boserif(0,0,8);

  kata.fu_stroke((220,580),(780,580),(340,20));
  if mincho>0.01:
    replace_strokep(0)((220,350)–(220,580)–oldp);
    replace_strokeq(0)((1.4,1.4)–(1.7,1.7)–oldq);
    set_botip(0,2,1);
    set_botip(0,4,0);
    set_boserif(0,2,whatever);
    set_boserif(0,4,4);
  else:
    replace_strokep(0)((220,350)–oldp);
    replace_strokeq(0)((1.4,1.4)–oldq);
    set_botip(0,1,1);
    set_botip(0,3,0);
```

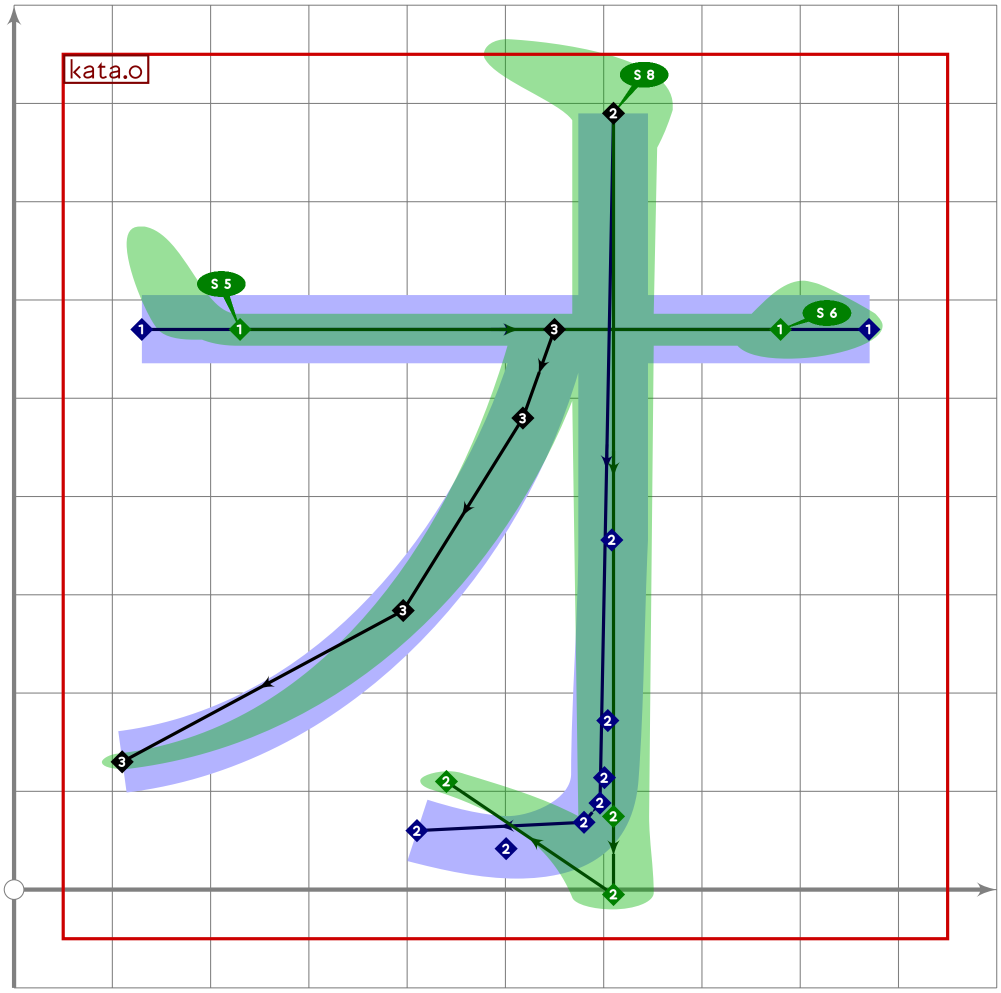
U+30A8
tsuku.uni30A8

```
    set_boserif(0,3,4);
  fi;
  set_boserif(0,0,whatever);
  set_boserif(0,1,8);
  expand_pbox;
enddef;
```



```

vardef kata.e =
  push_pbox_toexpand("kata.e");

  kata.ni;

  push_stroke((point 1 of get_strokep(-1))--(point 1 of get_strokep(0)),
    (1.5,1.5)--(1.5,1.5));
  expand_pbox;
enddef;
```

U+30AA
tsuku.uni30AA

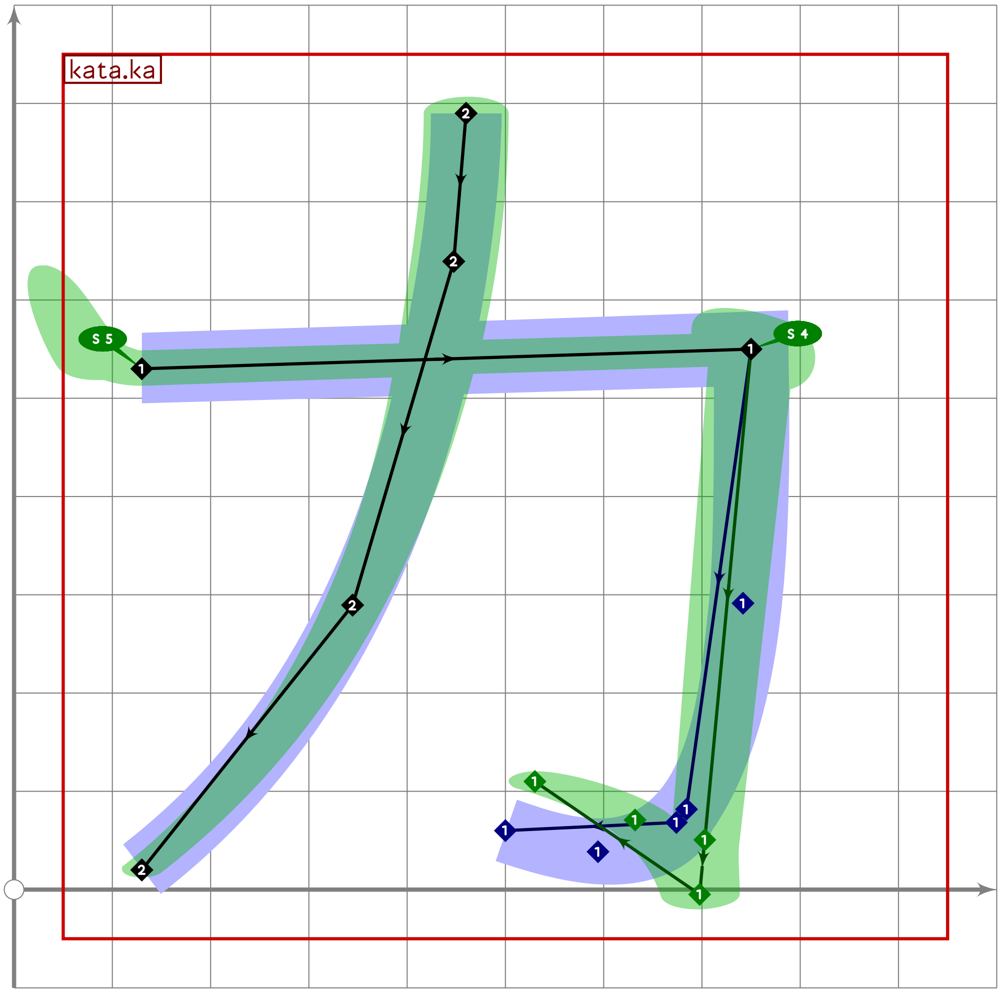


```

vardef kata.o =
  push_pbox_toexpand(”kata.o”);

  push_stroke((130+100*mincho,570)--(870-90*mincho,570),
    (1.8,1.8)--(1.6,1.6));
  set_boserif(0,0,5);
  set_boserif(0,1,6);

  kata.ho_centre((610,790),(610,20));

  kata.no_stroke((550,570),(110,130));
  expand_pbox;
enddef;

```

Katakana Kakikukeko/Gagigugego

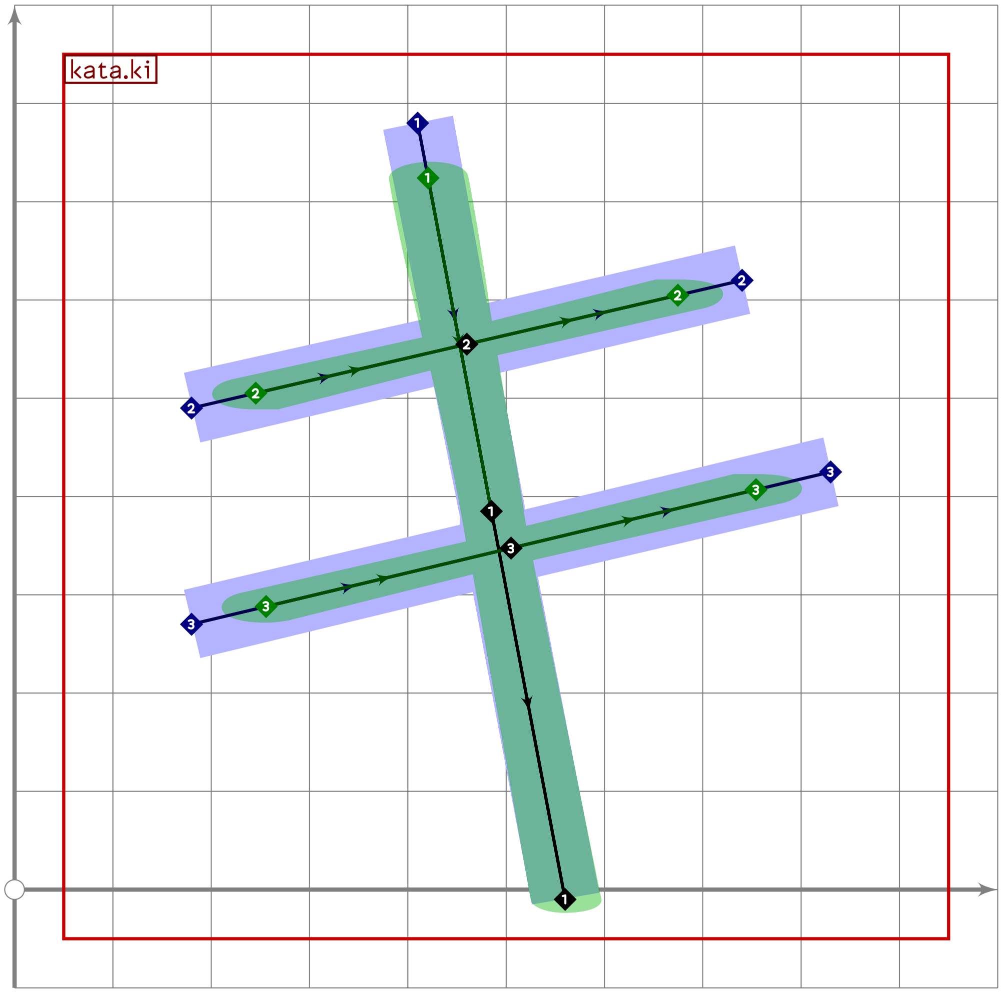
```
%%%%%%%%%%% KATAKANA KAKIKUKEKO/GAGIGUGEGO
```

U+30AB
tsuku.uni30AB



```
vardef kata.ka =
  push_pbox_toexpand(”kata.ka”);

  kata.ho_centre((750,550),(700,20));

  replace_strokep(0)((130,530)--oldp);
  replace_strokeq(0)((1.8,1.8)--oldq);
  set_botip(0,1,1);
  set_botip(0,2,whatever);
  set_botip(0,3,0);
  set_boserif(0,0,5);
  set_boserif(0,1,4);

  kata.no_stroke((460,790),(130,20));
  expand_pbox;
enddef;
```

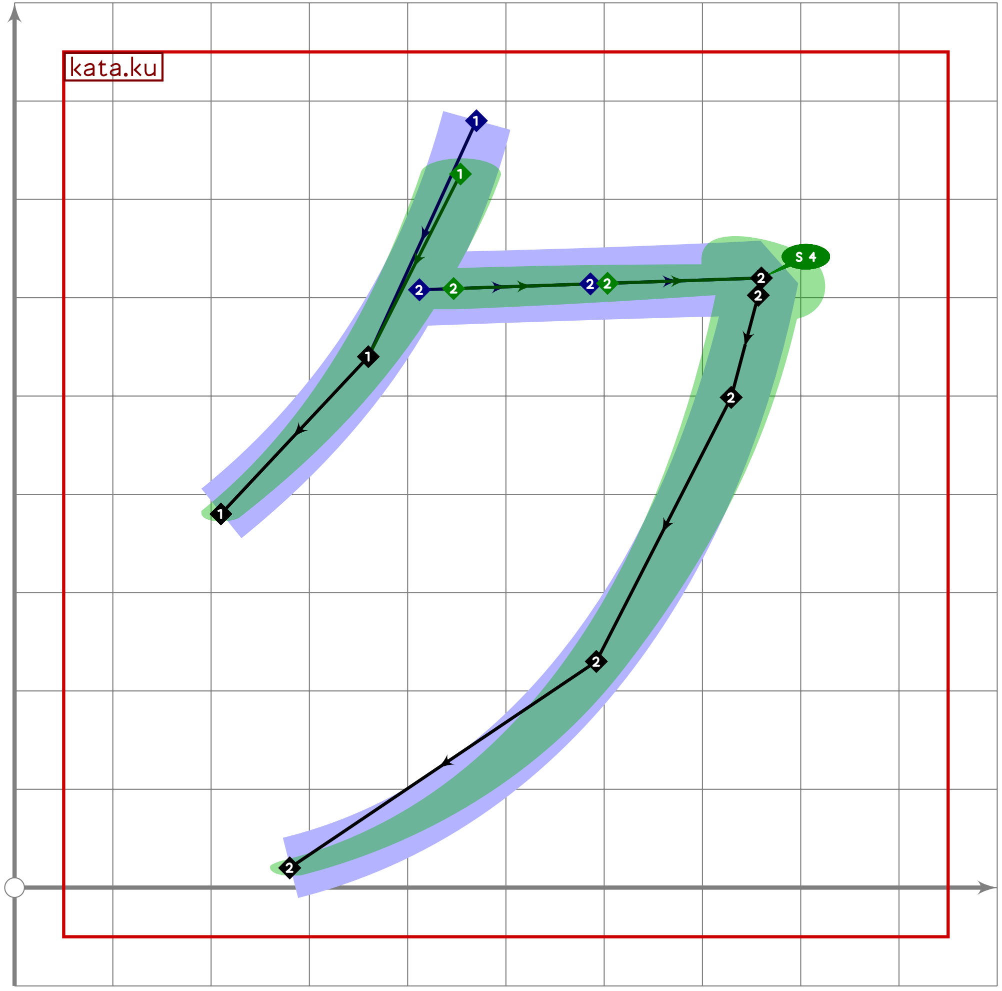
U+30AD
tsuku.uni30AD



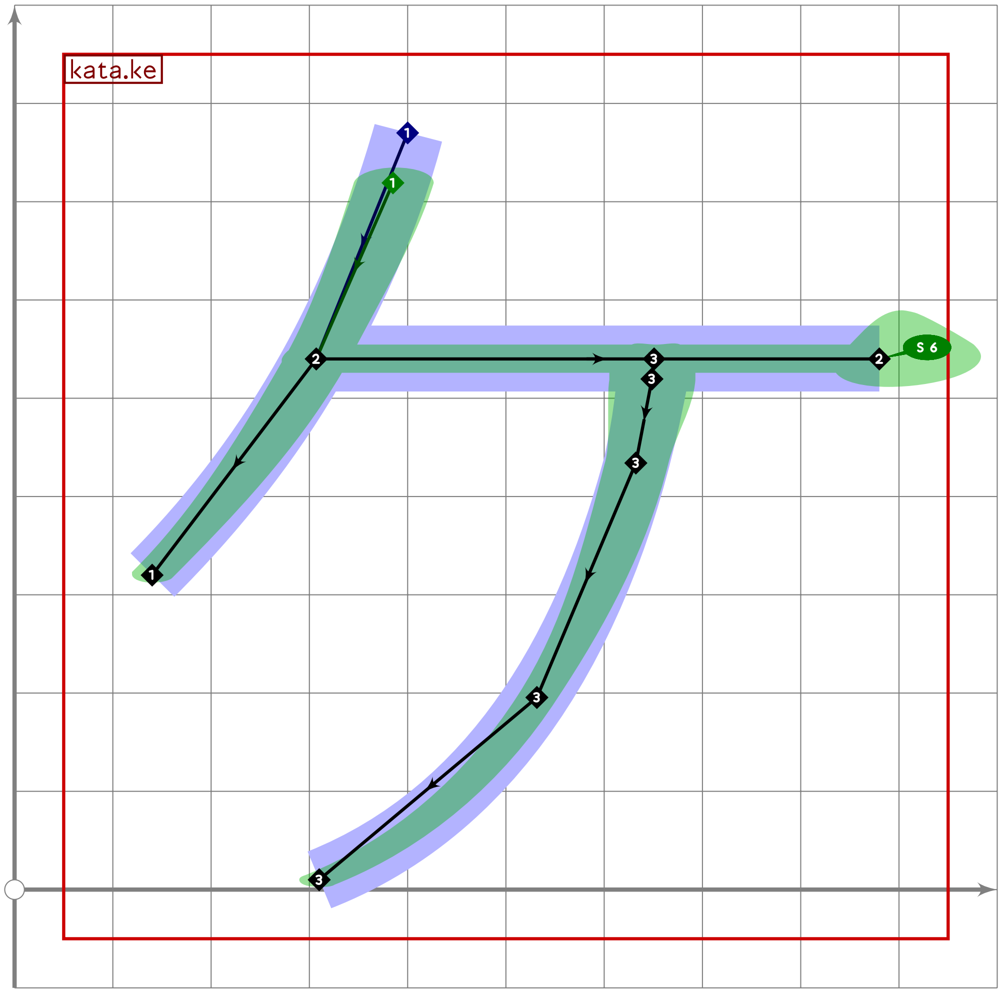
```

vardef kata.ki =
  push_pbox_toexpand(”kata.ki”);

  push_stroke((410,780)--(560,-10),
    (0.74,2.55)--(1.4,1.4)--(1.5,1.5));
  replace_strokep(0)(insert_nodes(oldp)(0.5));
  set_boserif(0,0,8);

  push_stroke((180,490)--(740,620),
    (0.6,3)--(1.6,1.6)--(0.6,3));
  replace_strokep(0)(insert_nodes(oldp)(0.5));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  push_stroke((180,270)--(830,425),
    (0.6,3)--(1.6,1.6)--(0.6,3));
  replace_strokep(0)(insert_nodes(oldp)(0.5));
  set_boserif(0,0,5);
  set_boserif(0,2,6);
  expand_pbox;
```

U+30AF
tsuku.uni30AF

```
enddef;
```

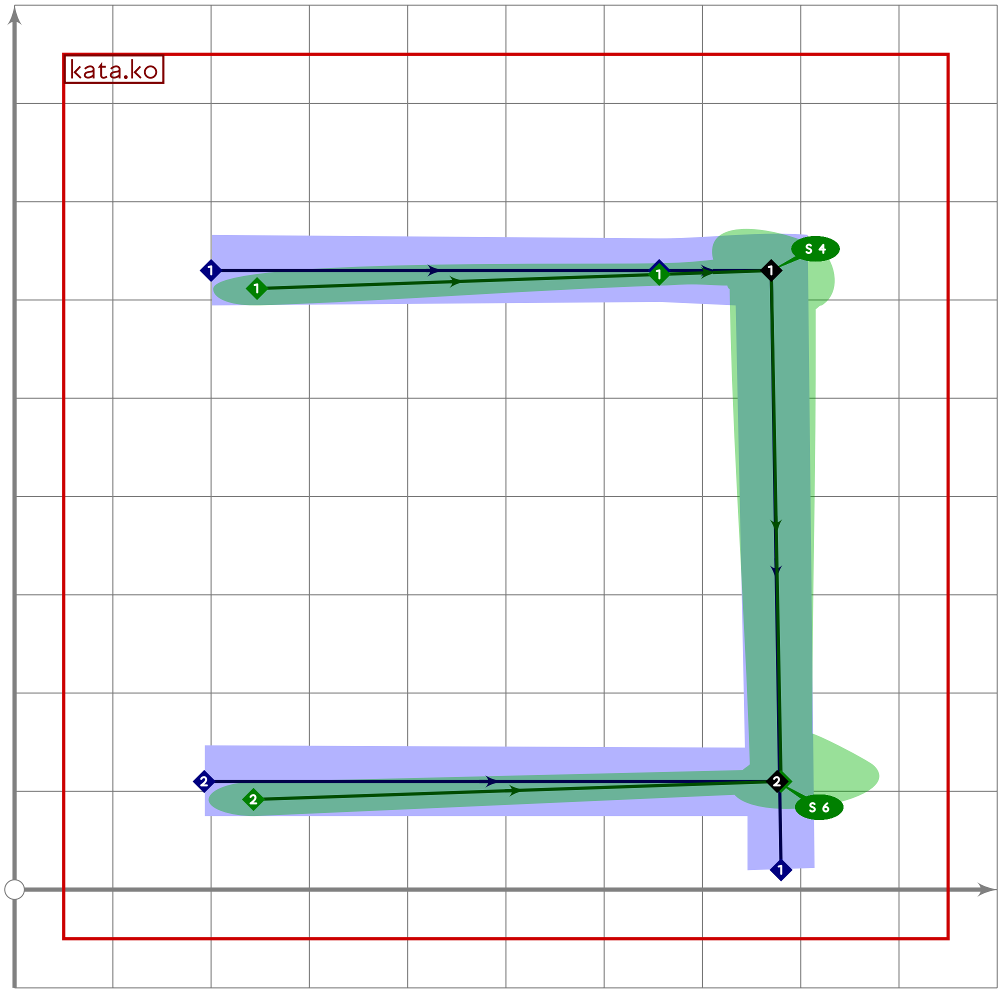


```

vardef  kata.ku =
  push_pbox_toexpand("kata.ku");

  push_stroke((470,780)..(360,540)..(210,380),
    (0.68,2.7)--(1.4,1.4)--(1.1,1.1));
  set_boserif(0,0,5);

  z1=(get_strokep(0) intersectionpoint ((0,600)--(1000,620)))+10*right;
  kata.fu_stroke(z1,(760,620),(280,20));
  set_boserif(0,0,whatever);
  expand_pbox;
enddef;
```



```
vardef kata.ke =
  push_pbox_toexpand(”kata.ke”);

  push_stroke((400,770)..(307,540)..(140,320),
    (0.68,2.7)–(1.4,1.4)–(1.1,1.1));
  set_boserif(0,0,5);

  z1=(get_strokep(0) intersectionpoint ((0,540)–(1000,540)));
  push_stroke(z1–(880,540),(1.5,1.5)–(1.5,1.5)–(0.75,2.85));
  set_boserif(0,1,6);

  kata.no_stroke(point 0.6 of (z1–(880,540)),(310,10));
  replace_strokep(0)(insert_nodes(oldp)(0.2));
  expand_pbox;
enddef;
```

U+30B3
tsuku.uni30B3

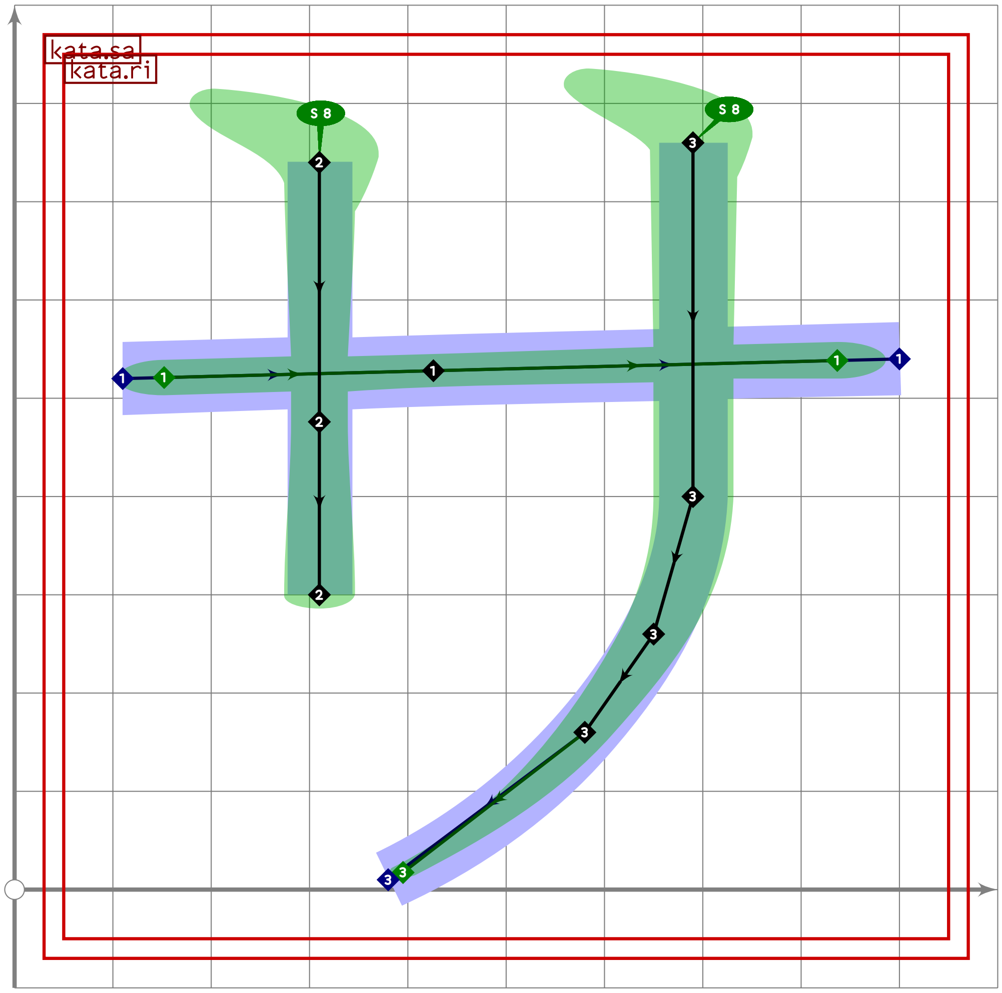


```

vardef kata.ko =
  push_pbox_toexpand(”kata.ko”);

  push_stroke((200,630-20*mincho)--(770,630)--(780,mincho[20,110]),
    (0.78,2.83)--(1.3,1.3)--(1.7,1.7)--(1.4,1.4));
  replace_strokep(0)(insert_nodes(oldp)(0.8));
  set_botip(0,2,1);
  set_boserif(0,0,5);
  set_boserif(0,2,4);

  push_stroke((193,110-20*mincho)--(776,110),
    (0.78,2.83)--(1.4,1.4));
  set_boserif(0,0,5);
  set_boserif(0,1,6);
  expand_pbox;
enddef;

```

Katakana Sashisuseso/Zajizuzezo

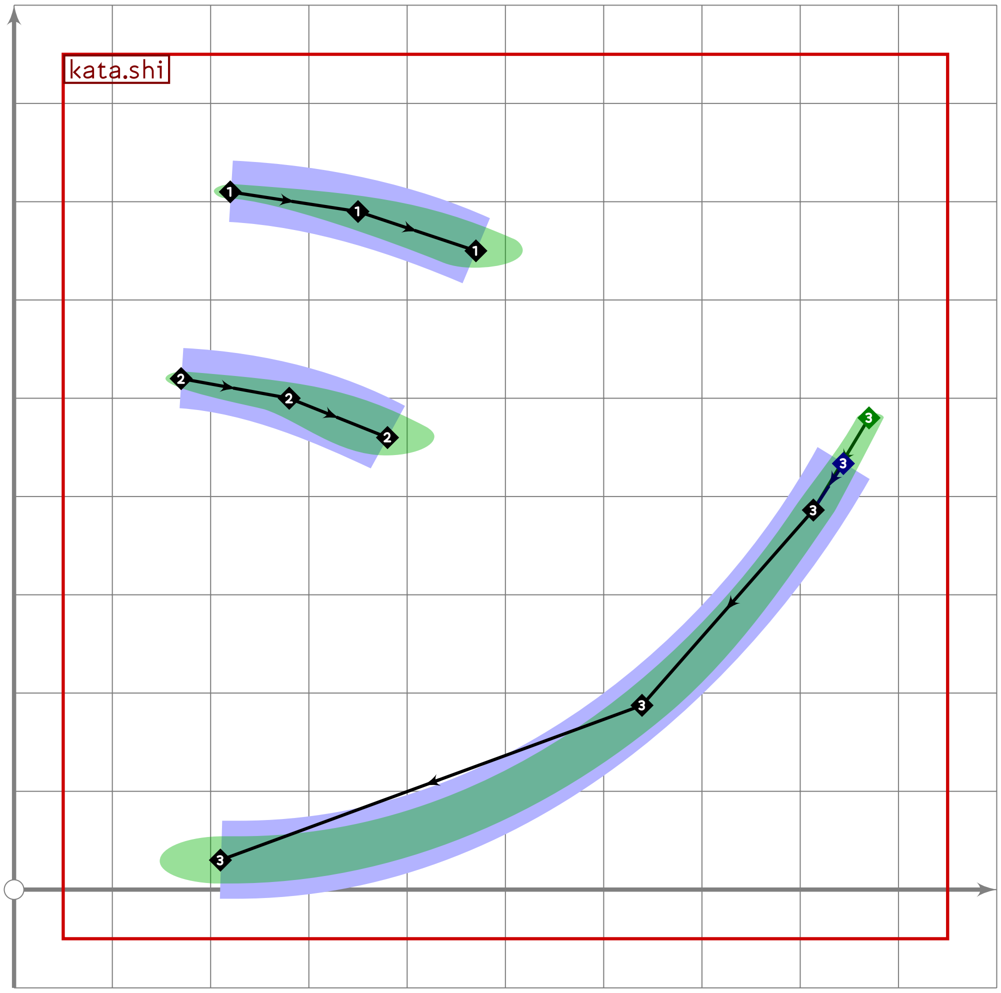
```
%%%%%%%%%% KATAKANA SASHISUSESO/ZAJIZUZEZO
```

U+30B5
tsuku.uni30B5



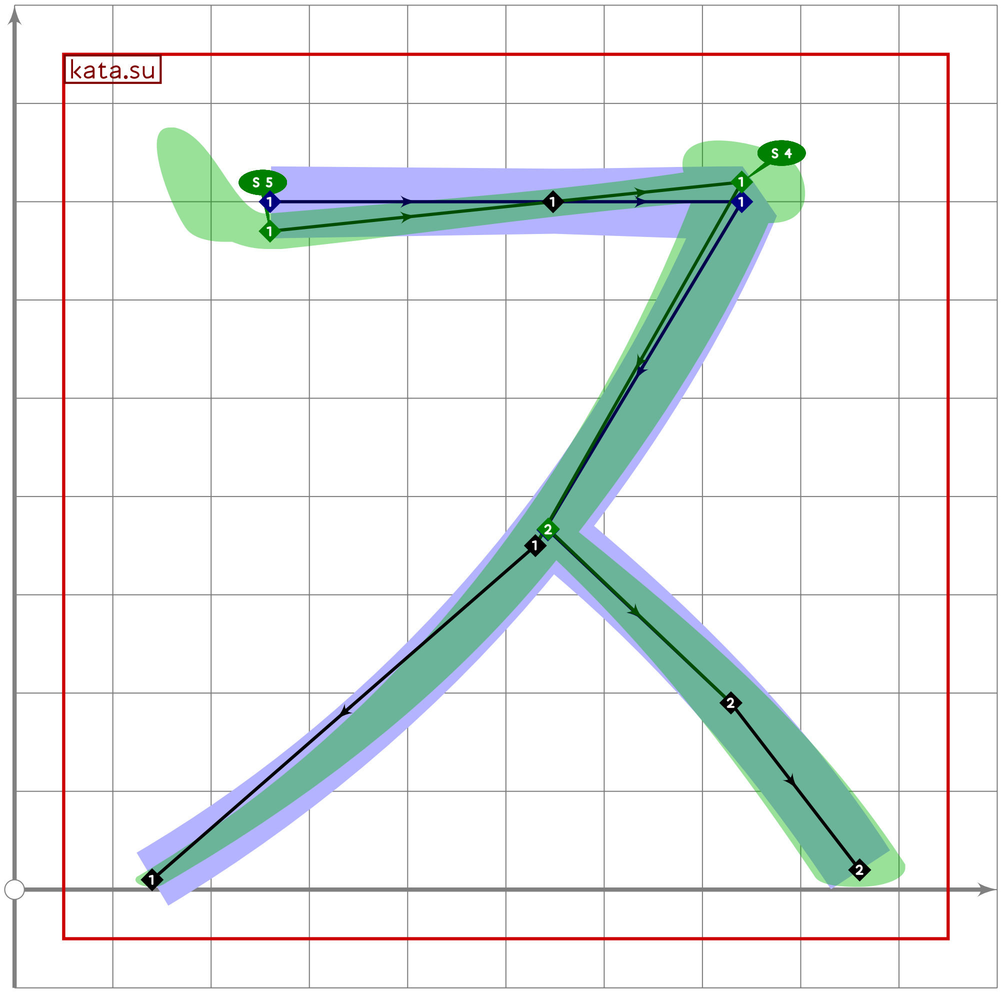
```

vardef  kata.sa  =
  push_pbox_toexpand(”kata.sa”);

  push_stroke((110,520)–(900,540),
    (0.7,3)–(1.7,1.7)–(0.7,3));
  replace_strokep(0)(insert_nodes(oldp)(0.4));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  kata.ri;
  expand_pbox;
enddef;
```

U+30B7
tsuku.uni30B7



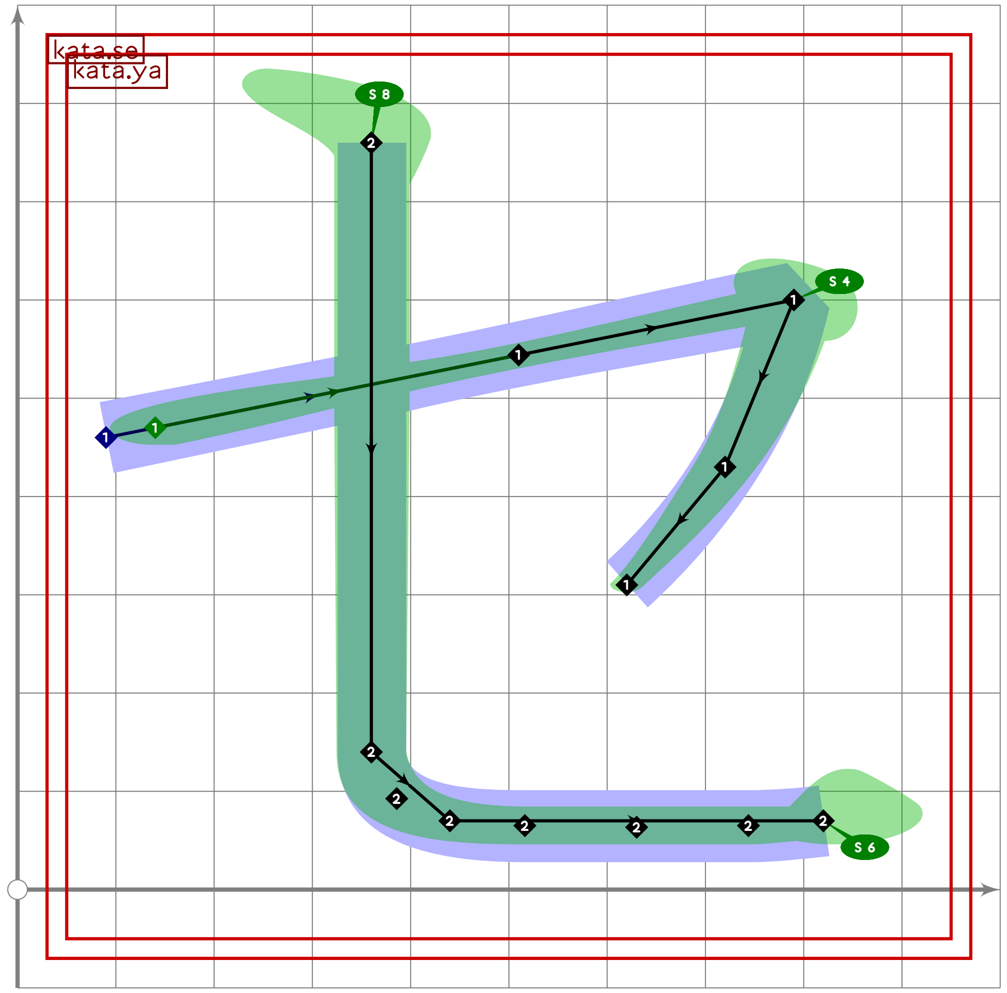
```
vardef  kata.shi =
  push_pbox_toexpand("kata.shi");

  push_stroke((220,710)..(350,690)..(470,650),
    (1,1)..(1.6,1.6)..(1.8,1.8));

  push_stroke((170,520)..(280,500)..(380,460),
    (1,1)..(1.6,1.6)..(1.8,1.8));

  kata.no_stroke((870,480),(210,30));

  replace_strokeq(0)((0.9,0.9)--(1.1,1.1)--(1.4,1.4)--(2.2,2.2));
  set_boserif(0,4,5);
  expand_pbox;
enddef;
```

U+30B9
tsuku.uni30B9



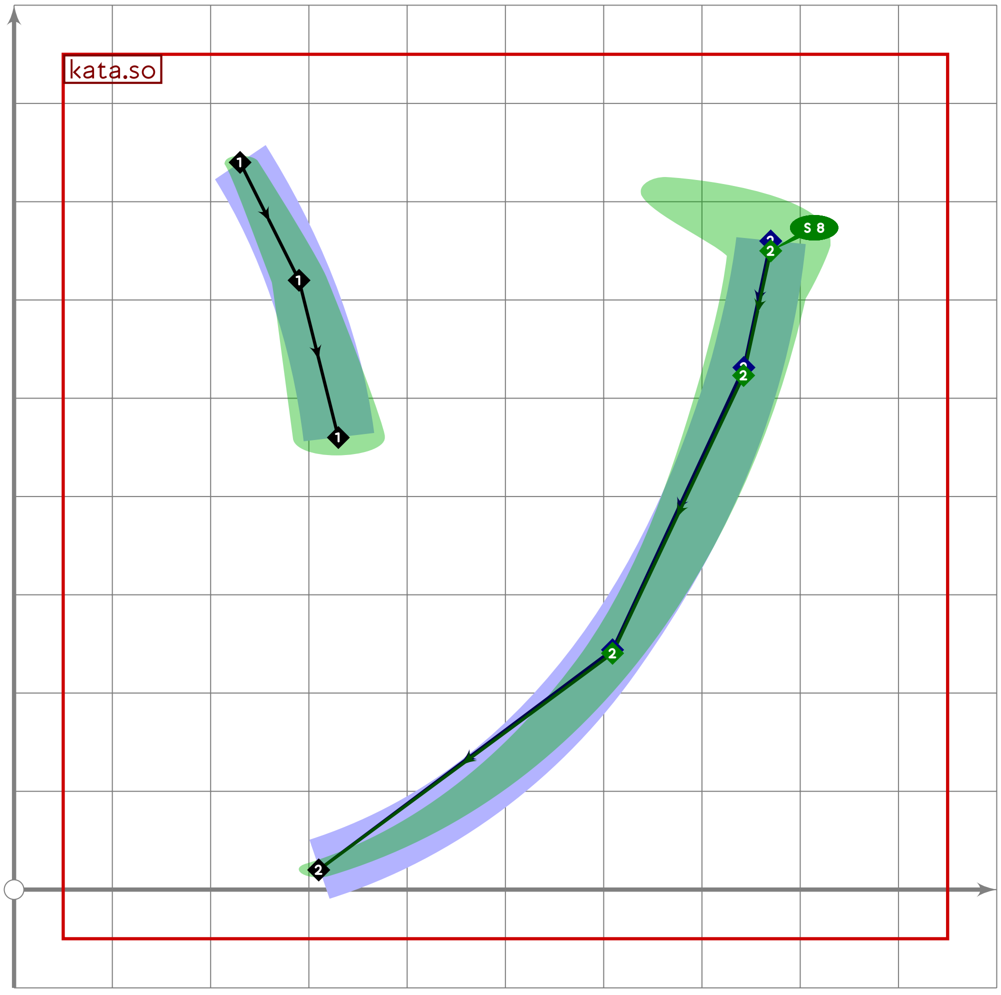
```
vardef kata.su =
  push_pbox_toexpand("kata.su");

  push_stroke((260,700-30*mincho)--(740,700+20*mincho)..(530,350)..(140,10),
    (1.8,1.8)--(1.3,1.3)--(1.7,1.7)--(1.4,1.4)--(1,1));
  replace_strokep(0)(insert_nodes(oldp)(0.6));
  set_botip(0,2,0);
  set_boserif(0,0,5);
  set_boserif(0,2,4);

  push_stroke((point 2.95 of get_strokep(0))..(729,190)..(860,20),
    (1.2,1.2)--(1.6,1.6)--(1.8,1.8));
  expand_pbox;
enddef;
```

U+30BB
tsuku.uni30BB



```
vardef kata.se =
  push_pbox_toexpand(”kata.se”);

  kata.ya;

  replace_strokep(-1)(oldp shifted (-30,0));

  replace_strokep(0)((360,760)–(360,140){dir 274}..
    (440,70)..tension 2.1..(820,70));
  replace_strokeq(0)((1.6,1.6)–(1.5,1.5)–(1.9,1.9)–(1.8,1.8));
  set_boserif(0,0,8);
  set_boserif(0,3,6);
  expand_pbox;
enddef;
```

U+30BD
tsuku.uni30BD

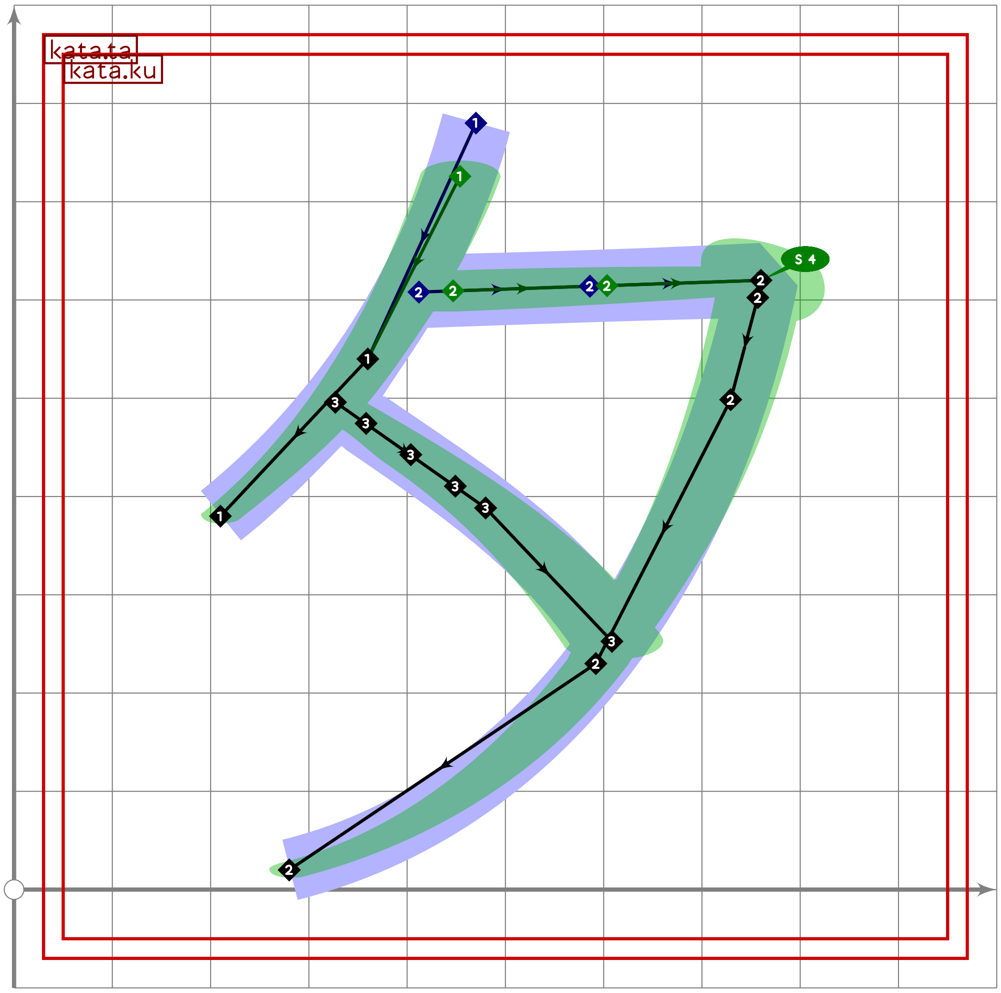


```
vardef kata.so =
  push_pbox_toexpand(”kata.so”);

  push_stroke((230,740)..(290,620)..(330,460),
    (1,1)..(1.3,1.3)..(1.8,1.8));

  kata.no_stroke((770,660-10*mincho),(310,20));
  set_boserif(0,0,8);
  expand_pbox;
enddef;

```

Katakana Tachitsuteto/Dajizudedo

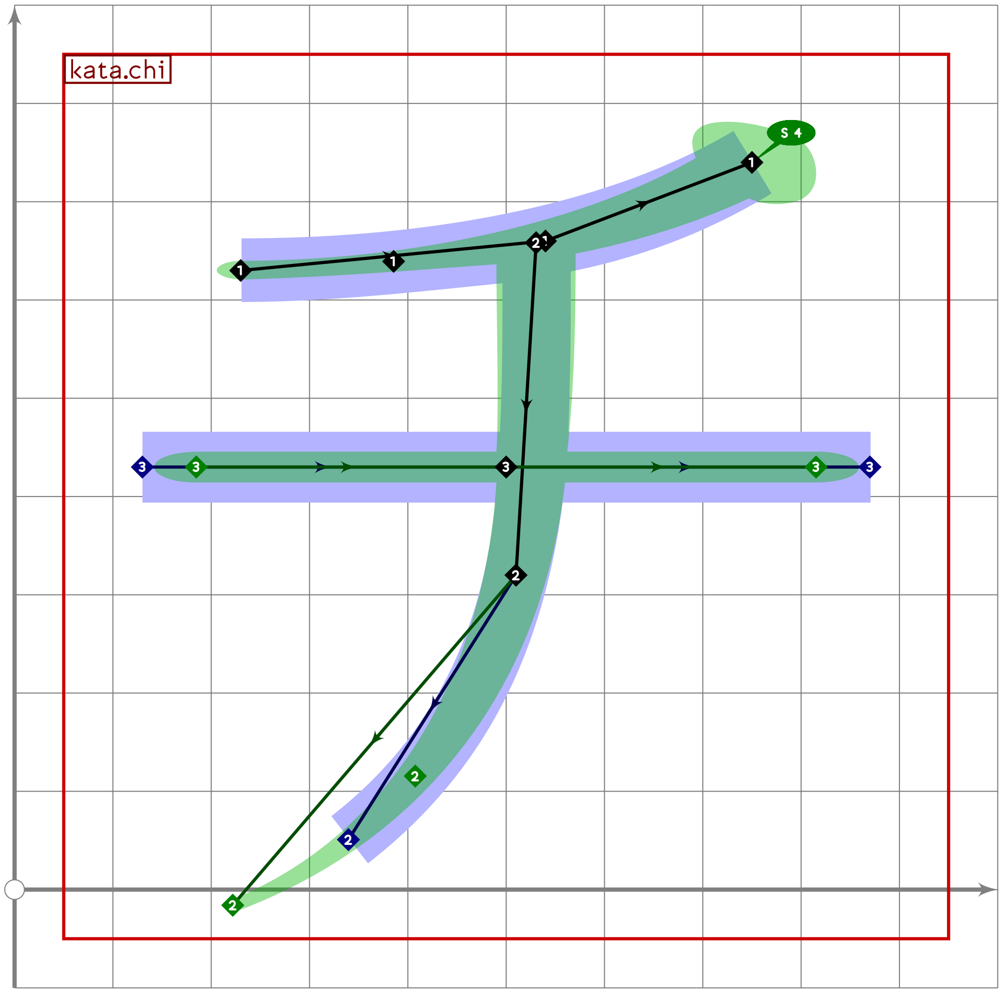
```
%%%%%%%%%% KATAKANA TACHITSUTETO/DAJIZUDEDO
```

U+30BF
tsuku.uni30BF



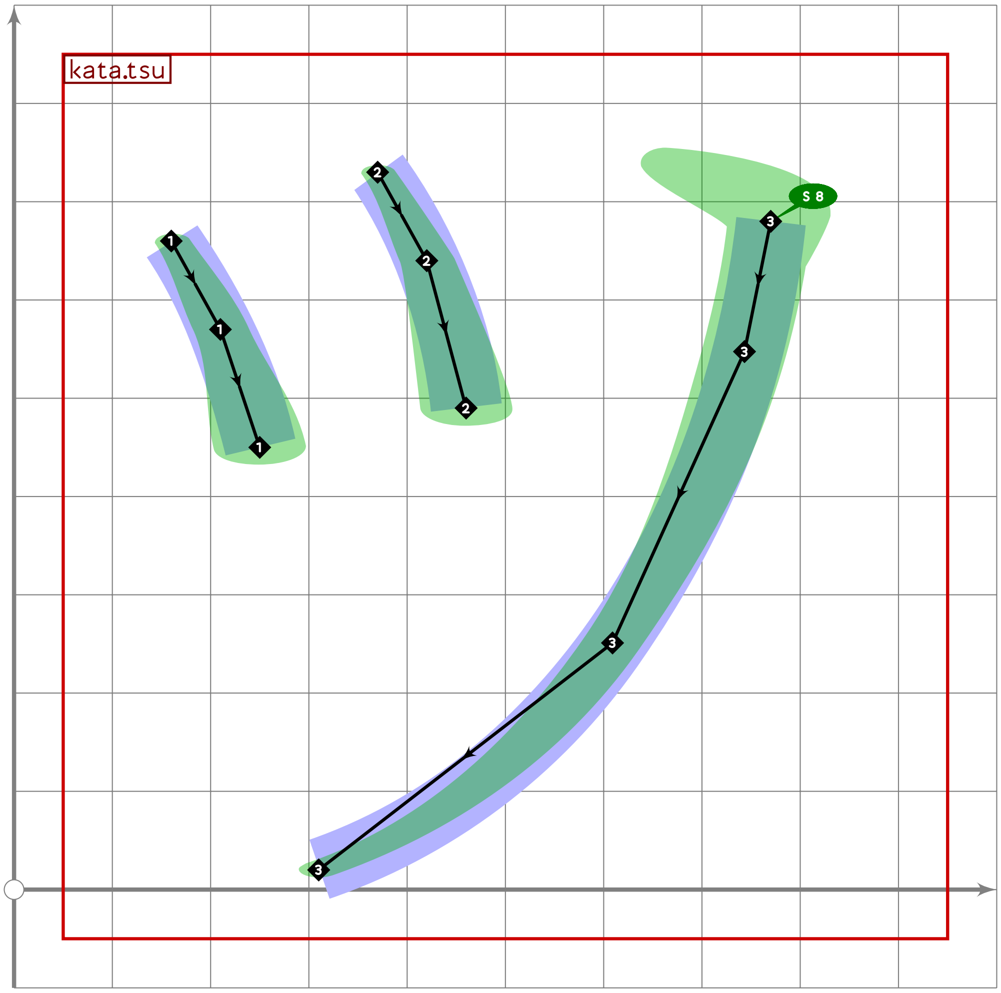
```
vardef kata.ta =
  push_pbox_toexpand("kata.ta");

  kata.ku;

  numeric x[],y[];
  z1=point 1.25 of get_strokep(-1);
  z3=point 4.9 of get_strokep(0);
  z2=(0.5[z1,z3])+0.05*((z3-z1) rotated 90);
  push_stroke(z1..tension 2..z2..z3,
    (1.2,1.2)--(1.6,1.6)--(1.9,1.9));
  expand_pbox;
enddef;
```

U+30C1
tsuku.uni30C1

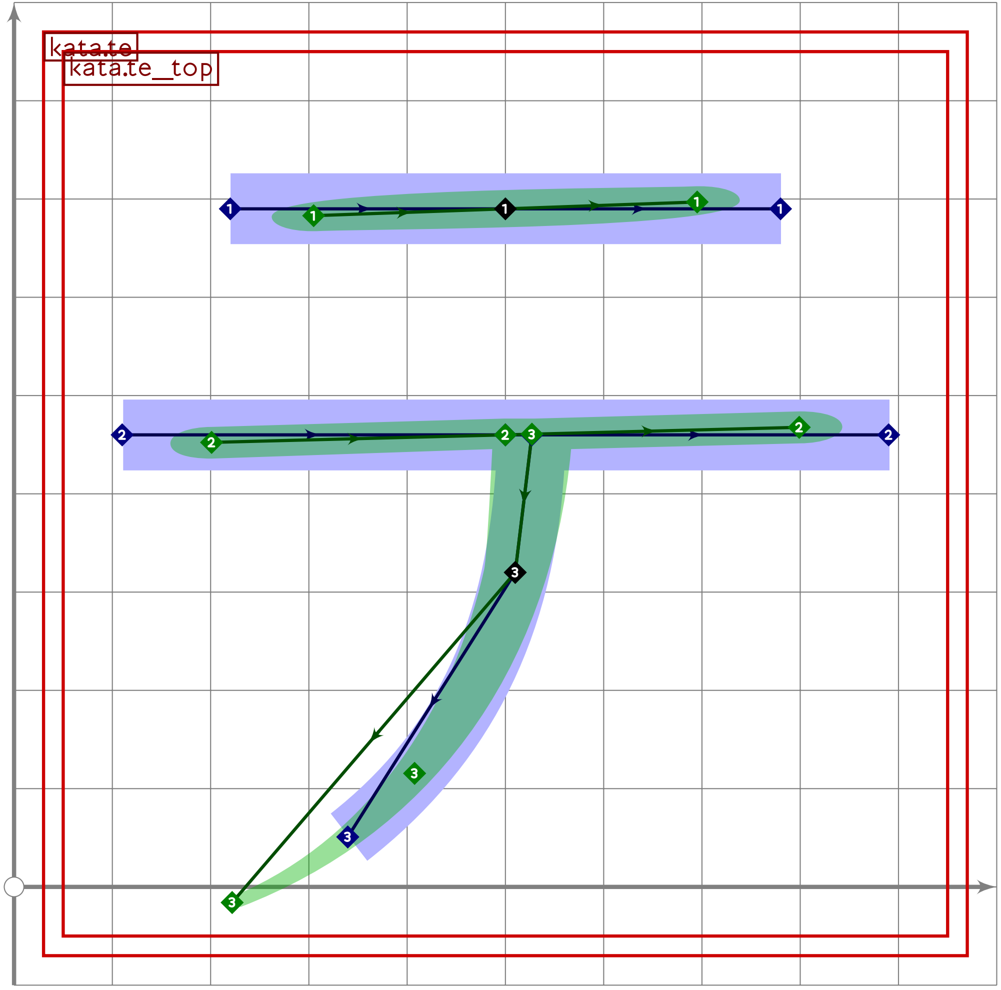


```

vardef kata.chi =
  push_pbox_toexpand(”kata.chi”);

  push_stroke((230,630)..tension 1.3..(540,660)..(750,740),
    (1.2,1.2)–(1.7,1.7)–(2,2));
  set_boserif(0,2,4);

  kata.na_centre;
  replace_strokep(0)(subpath (xpart (oldp intersectiontimes
    get_strokep(-1)),infinity) of oldp);

  push_stroke((130,430)–(870,430),
    (0.7,2.7)–(1.6,1.6)–(0.7,2.7));
  replace_strokep(0)(insert_nodes(oldp)(0.5));
  set_boserif(0,0,5);
  set_boserif(0,2,6);
  expand_pbox;
enddef;
```

U+30C4
tsuku.uni30C4

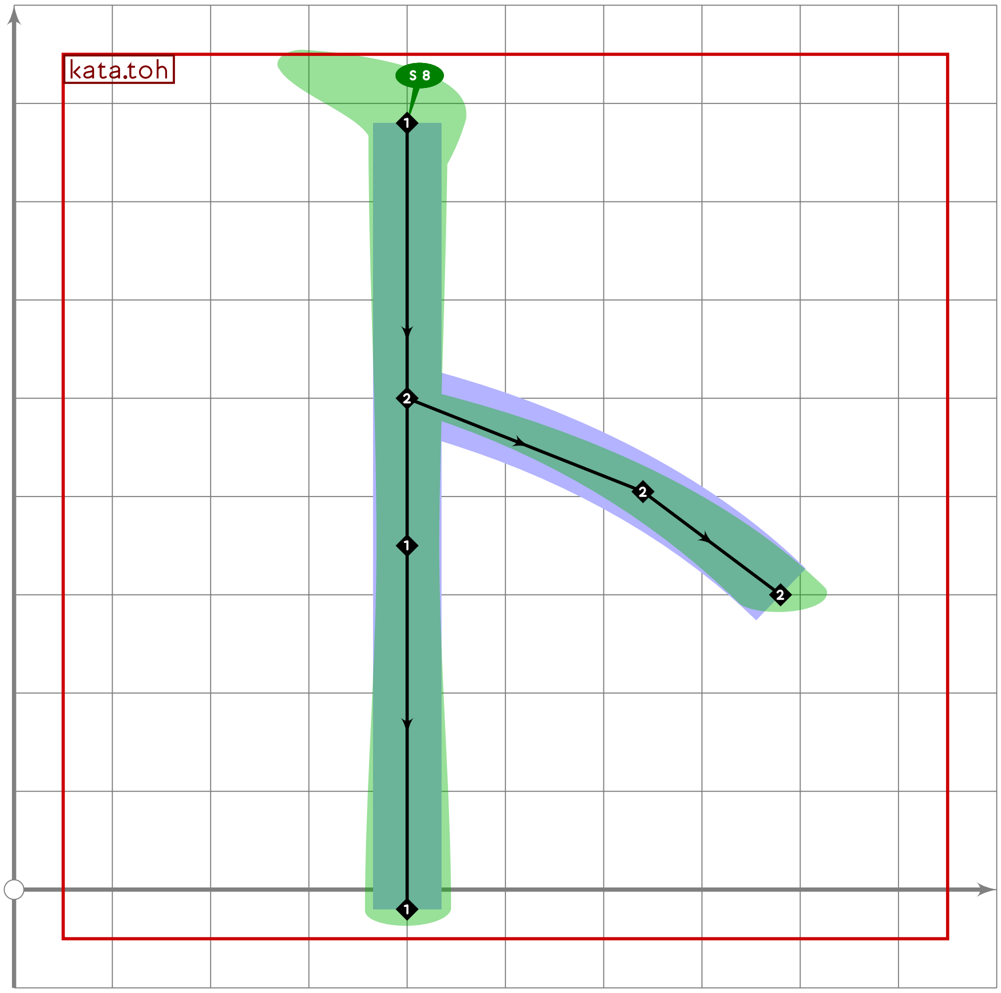


```

vardef  kata.tsu  =
  push_pbox_toexpand("kata.tsu");

  push_stroke((160,660)..(210,570)..(250,450),
    (1,1)..(1.3,1.3)..(1.8,1.8));

  push_stroke((370,730)..(420,640)..(460,490),
    (1,1)..(1.3,1.3)..(1.8,1.8));

  kata.no_stroke((770,680),(310,20));
  set_boserif(0,0,8);
  expand_pbox;
enddef;

vardef  kata.te_top  =
  push_pbox_toexpand("kata.te_top");

  push_stroke((220,690-10*mincho)--(780,690+10*mincho),
    (0.5,2.9)--(1.6,1.6)--(0.5,2.9));
  replace_strokep(0)(insert_nodes(oldp)(0.5));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  push_stroke((110,460-10*mincho)--(890,460+10*mincho),
    (0.6,2.8)--(1.6,1.6)--(0.6,2.8));
  replace_strokep(0)(insert_nodes(oldp)(0.5));
  set_boserif(0,0,5);
  set_boserif(0,2,6);
  expand_pbox;
enddef;
```



```

vardef kata.te =
  push_pbox_toexpand("kata.te");

  kata.te_top;

  kata.na_centre;
  replace_strokep(0)(subpath (xpart (oldp intersectiontimes get_strokep(-1)),
    infinity) of oldp);
  expand_pbox;
enddef;
```

U+30C8
tsuku.uni30C8

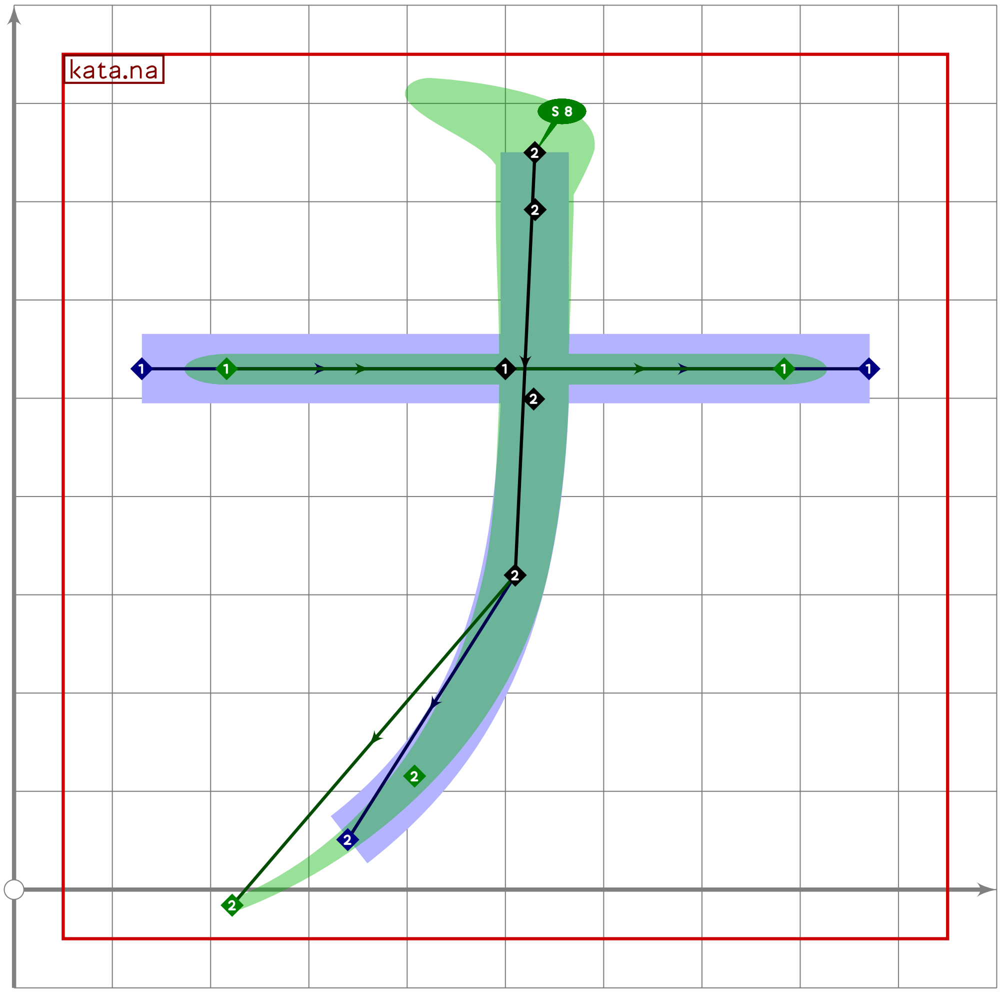


```

vardef kata.toh =
  push_pbox_toexpand(”kata.toh”);

  push_stroke((400,780)--(400,350)--(400,-20),
    (1.6,1.6)--(1.4,1.4)--(1.7,1.7));
  set_boserif(0,0,8);

  push_stroke((400,500)..tension 1.1..(640,405)..(780,300),
    (1.3,1.3)--(1.6,1.6)--(1.8,1.8));
  expand_pbox;
enddef;

```

Katakana Naninuneno

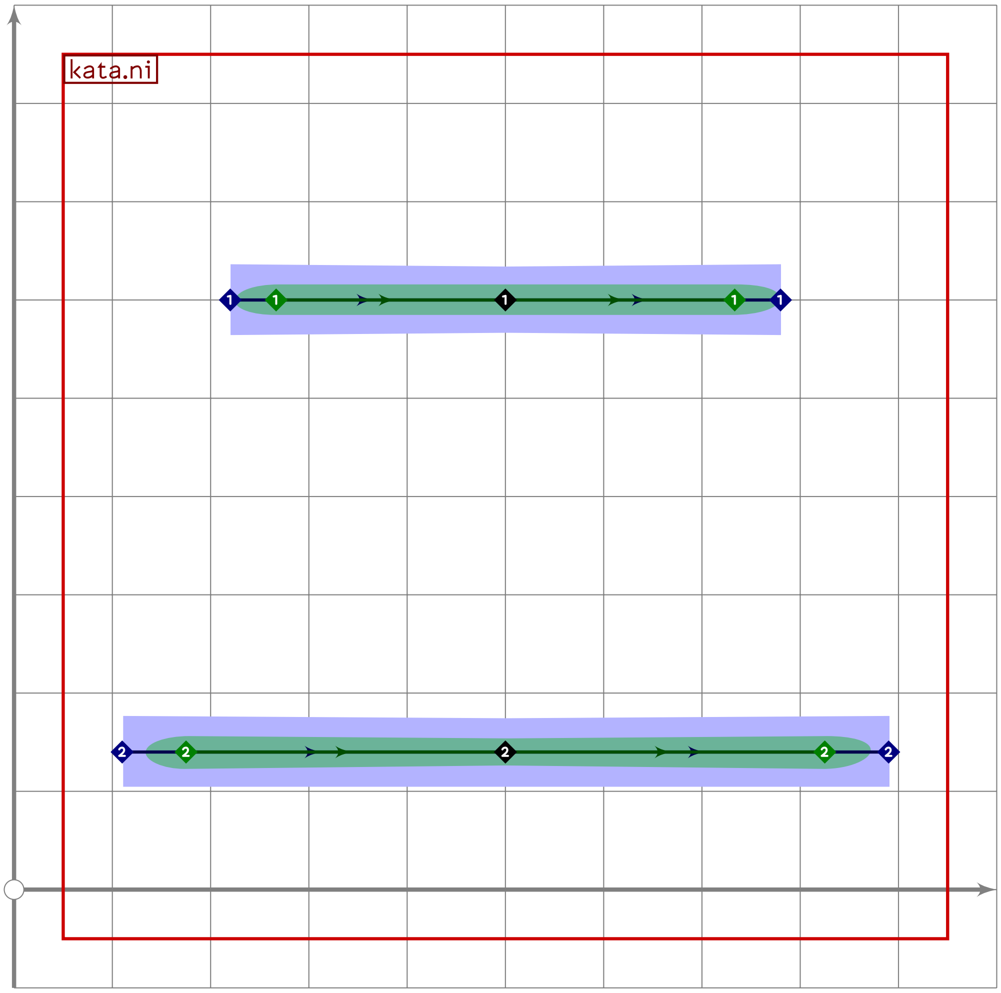
```
%%%%%%%%%% KATAKANA NANINUNENO

vardef kata.na_centre =
  push_stroke((530,750){down}..tension 1.2..(510,320)..(180,-30),
    (1.6,1.6)--(1.4,1.4)--(0.78,0.78));
enddef;
```

U+30CA
tsuku.uni30CA



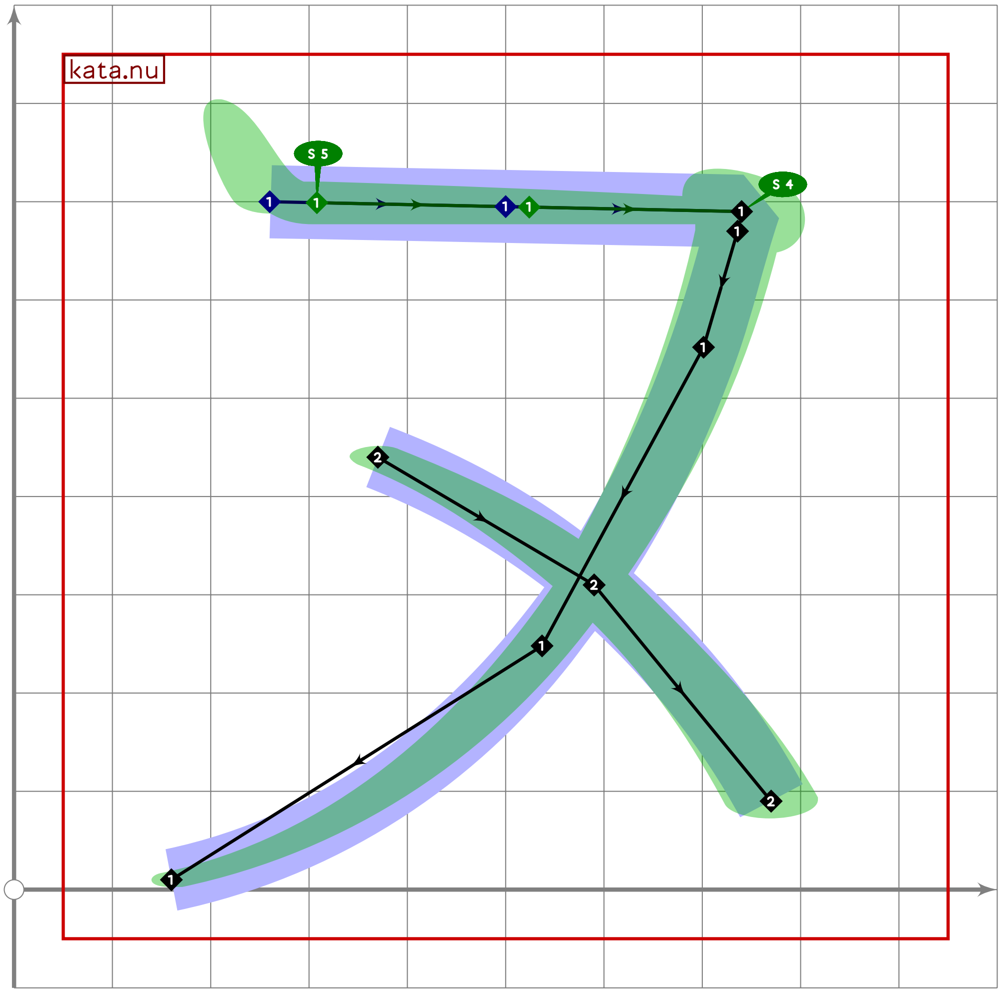
```

vardef kata.na =
  push_pbox_toexpand(”kata.na”);

  push_stroke((130,530)−(870,530),
    (0.6,2.8)−(1.6,1.6)−(0.6,2.8));
  replace_strokep(0)(insert_nodes(oldp)(0.5));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  kata.na_centre;
  set_boserif(0,0,8);
  expand_pbox;
enddef;
```

U+30CB
tsuku.uni30CB



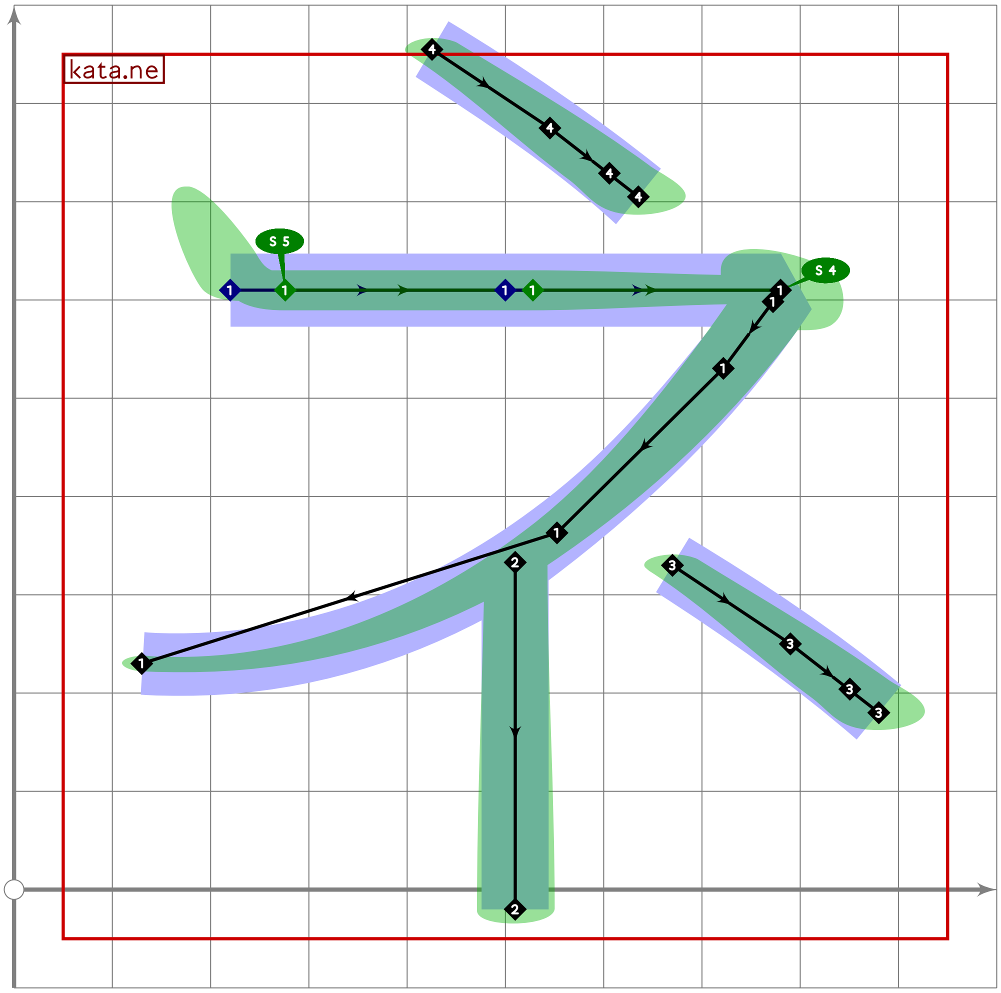
```
vardef  kata.ni =
  push_pbox_toexpand("kata.ni");

  push_stroke((220,600)--(500,600)--(780,600),
    (0.7,2.7)--(1.5,1.5)--(0.7,2.9));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  push_stroke((110,140)--(500,140)--(890,140),
    (0.7,2.7)--(1.5,1.5)--(0.7,2.9));
  set_boserif(0,0,5);
  set_boserif(0,2,6);
  expand_pbox;
enddef;
```

U+30CC
tsuku.uni30CC

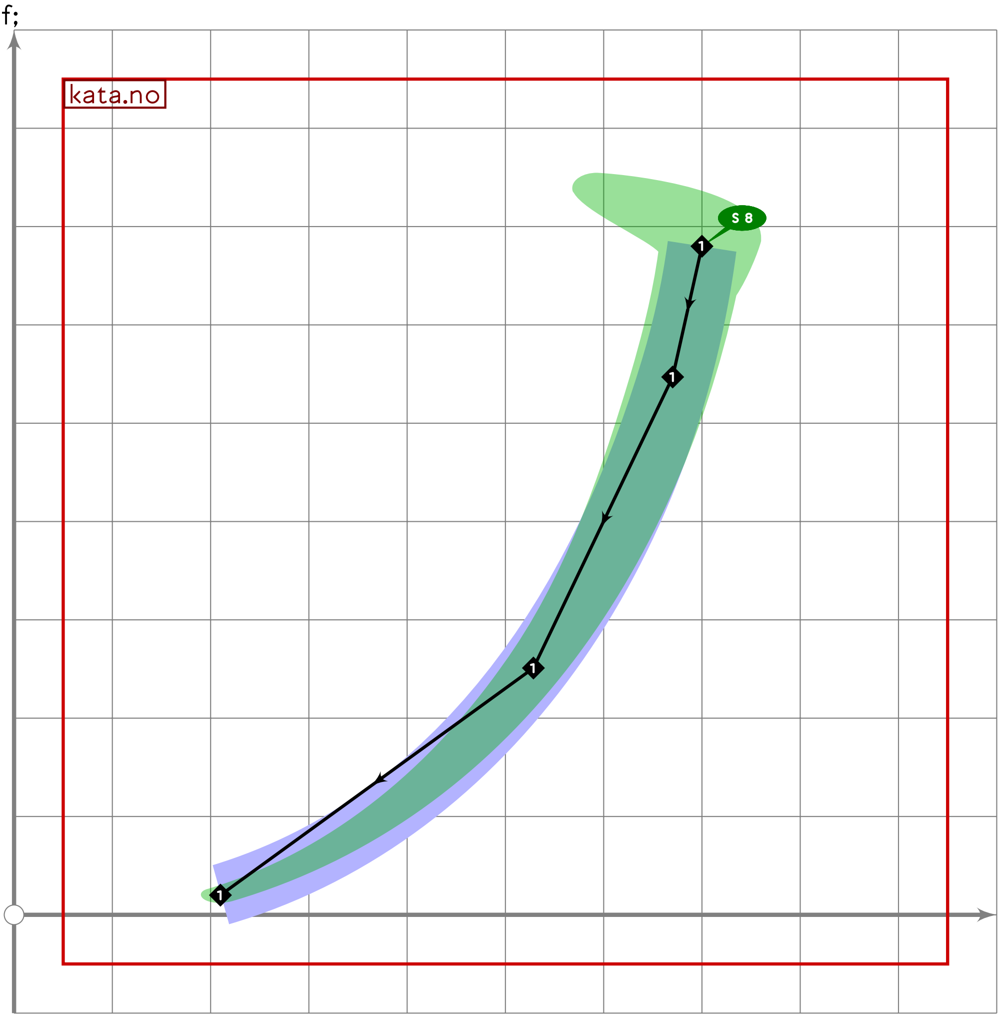


```

vardef  kata.nu =
  push_pbox_toexpand("kata.nu");

  kata.fu_stroke((260,700),(740,690),(160,10));

  push_stroke((370,440)..(590,310)..(770,90),
    (1.3,1.3)--(1.6,1.6)--(1.8,1.8));
  expand_pbox;
enddef;
```

U+30CD
tsuku.uni30CD



```
vardef kata.ne =
  push_pbox_toexpand(”kata.ne”);

  kata.fu_stroke((220,610),(780,610),(130,230));

  push_stroke((((510,0)--(510,1000)) intersectionpoint
      reverse get_strokep(0))--(510,-20),
    (1.4,1.4)--(1.6,1.6));

  push_stroke((670,330)..(790,250)..(880,180),
    (1.3,1.3)--(1.6,1.6)--(1.8,1.8));

  push_stroke(get_strokep(0) shifted ((510,800)-point 0.7 of get_strokep(0)),
    get_strokeq(0));
  expand_pbox;
enddef;

vardef kata.no_stroke(expr ur,ll) =
  push_stroke(insert_nodes(ur..tension 1.1..
      (0.65[xpart ll,xpart ur],0.35[ypart ll,ypart ur])..{curl 1.2}ll)(0.3),
      (1.7,1.7)--(1.7,1.7)--(1.4,1.4)--(1.1,1.1));
enddef;
```

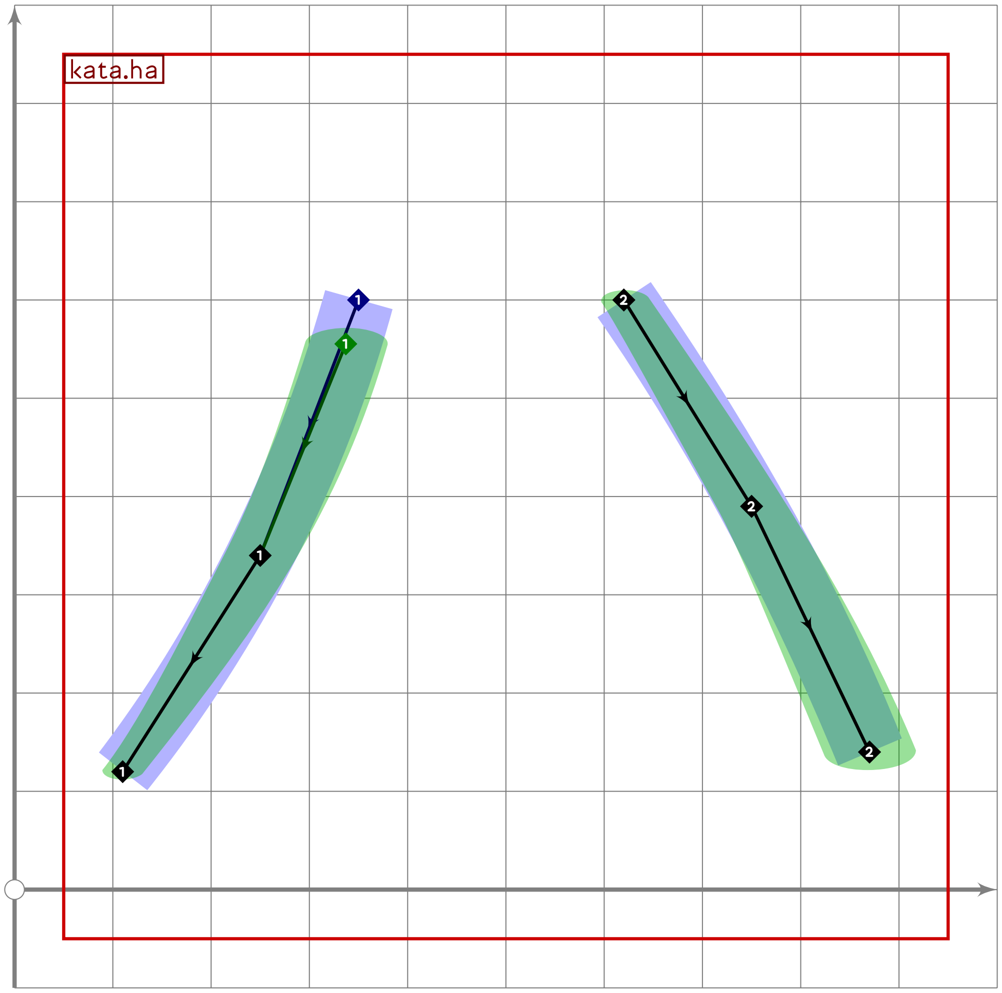


```

vardef kata.no =
  push_pbox_toexpand("kata.no");

  kata.no_stroke((700,680),(210,20));
  set_boserif(0,0,8);
  expand_pbox;
enddef;

```

Katakana Hahifuheho/Babibubebo/Papipupepo

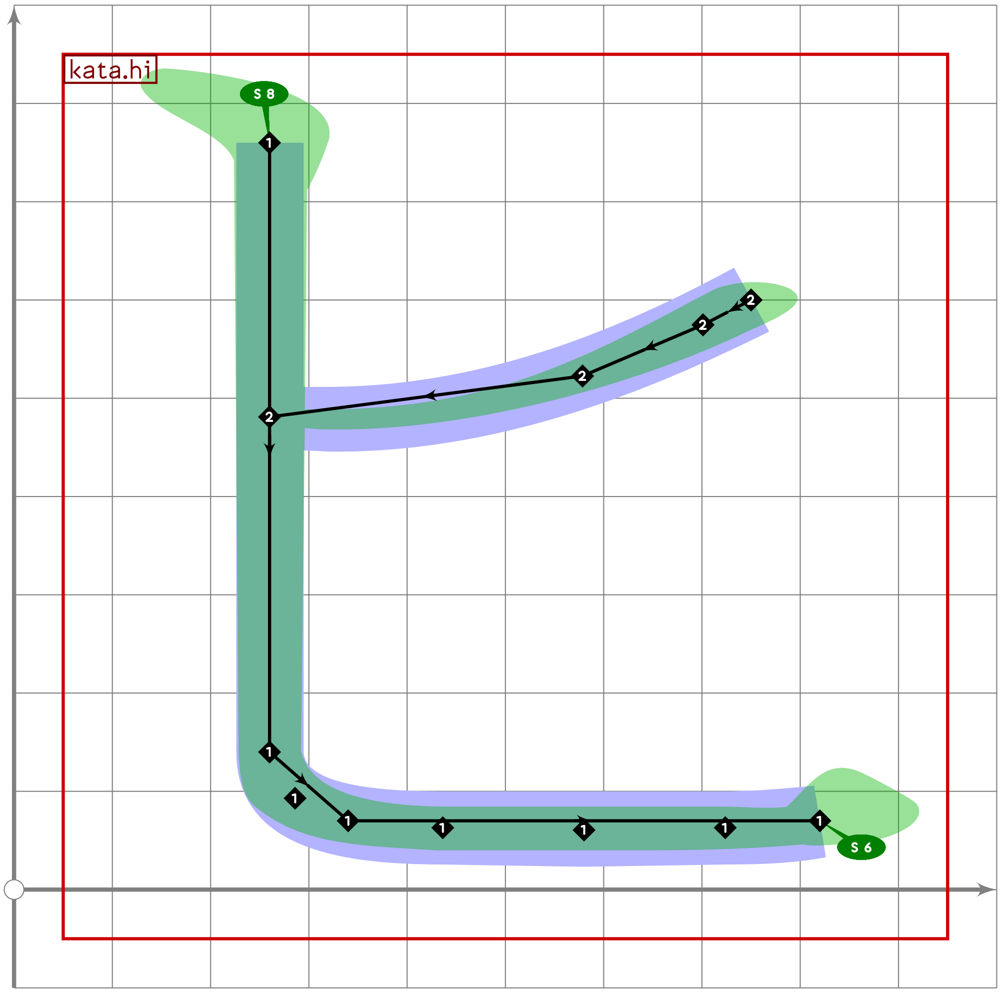
```
%%%%%%%%%% KATAKANA HAHIFUHEHO/BABIBUBEBO/PAPIPUPEPO
```

U+30CF
tsuku.uni30CF



```

vardef  kata.ha =
  push_pbox_toexpand(”kata.ha”);

  push_stroke((350,600)..(250,340)..(110,120),
    (0.7,2.7)–(1.5,1.5)–(1.1,1.1));
  set_boserif(0,0,8);

  push_stroke((620,600)..(750,390)..(870,140),
    (1.2,1.2)–(1.5,1.5)–(1.8,1.8));
  expand_pbox;
enddef;
```

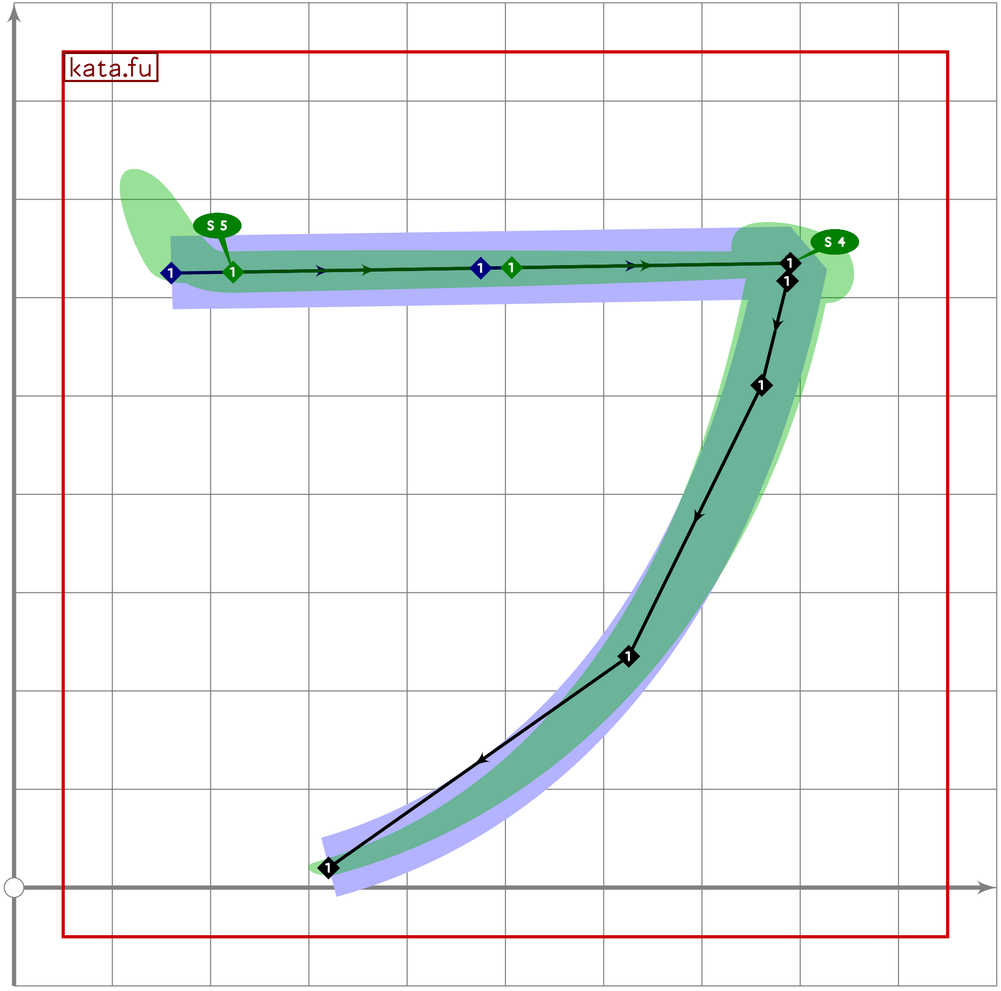
U+30D2
tsuku.uni30D2



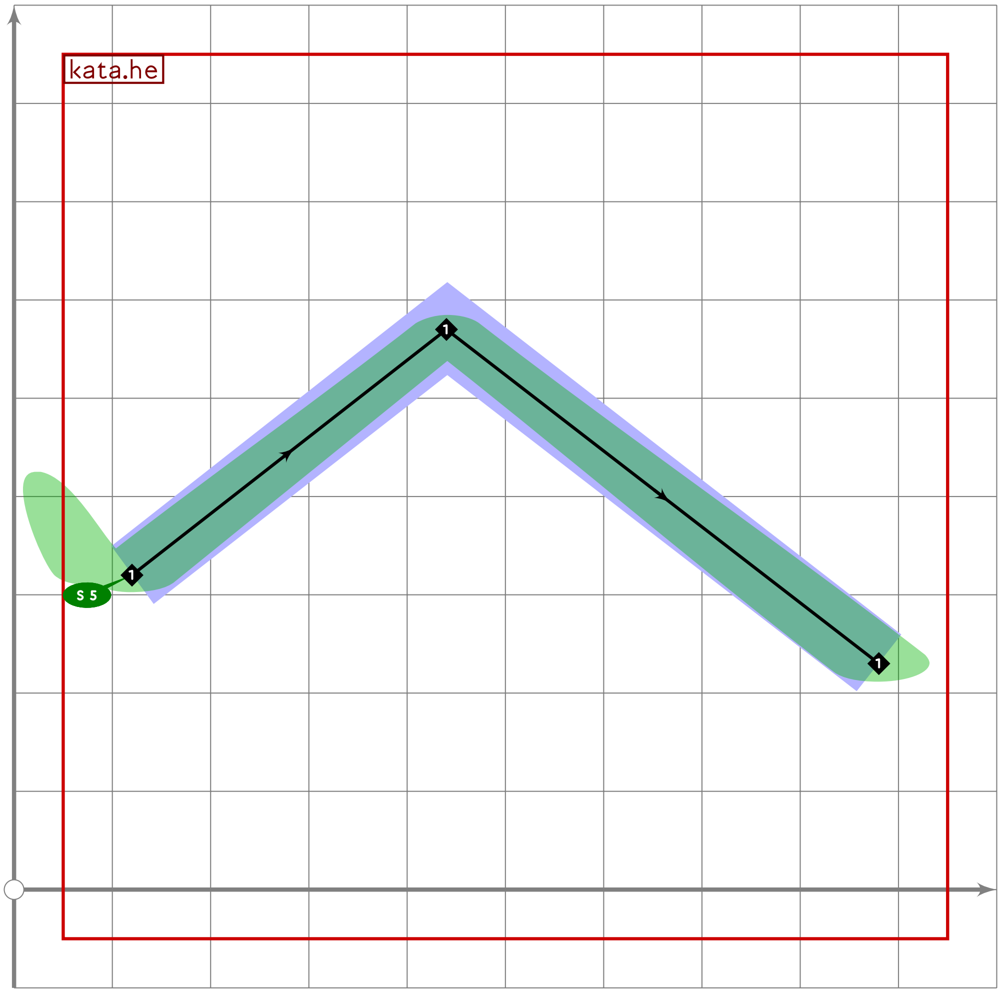
```

vardef kata.hi =
  push_pbox_toexpand("kata.hi");

  push_stroke((260,760)--(260,140){dir 274}..(340,70)..tension 2.1..(820,70),
    (0.84,2.18)--(1.4,1.4)--(2.1,2.1)--(1.9,1.9));
  set_boserif(0,0,8);
  set_boserif(0,3,6);

  kata.no_stroke((750,600),point 0.45 of get_strokep(0));
  replace_strokeq(0)(oldq shifted (0.1,0.1));
  expand_pbox;
enddef;

vardef kata.fu_stroke(expr ul,ur,ll) =
  kata.no_stroke(ur,ll);
  replace_strokep(0)(insert_nodes(((mincho*0.1)[ul,ur])--oldp)(0.5,1.15));
  replace_strokeq(0)((2,2)--(1.9,1.9)--(1.5,1.5)--oldq);
  set_botip(0,2,0);
  set_boserif(0,0,5);
  set_boserif(0,2,4);
```

U+30D5
tsuku.uni30D5

```
enddef;
```



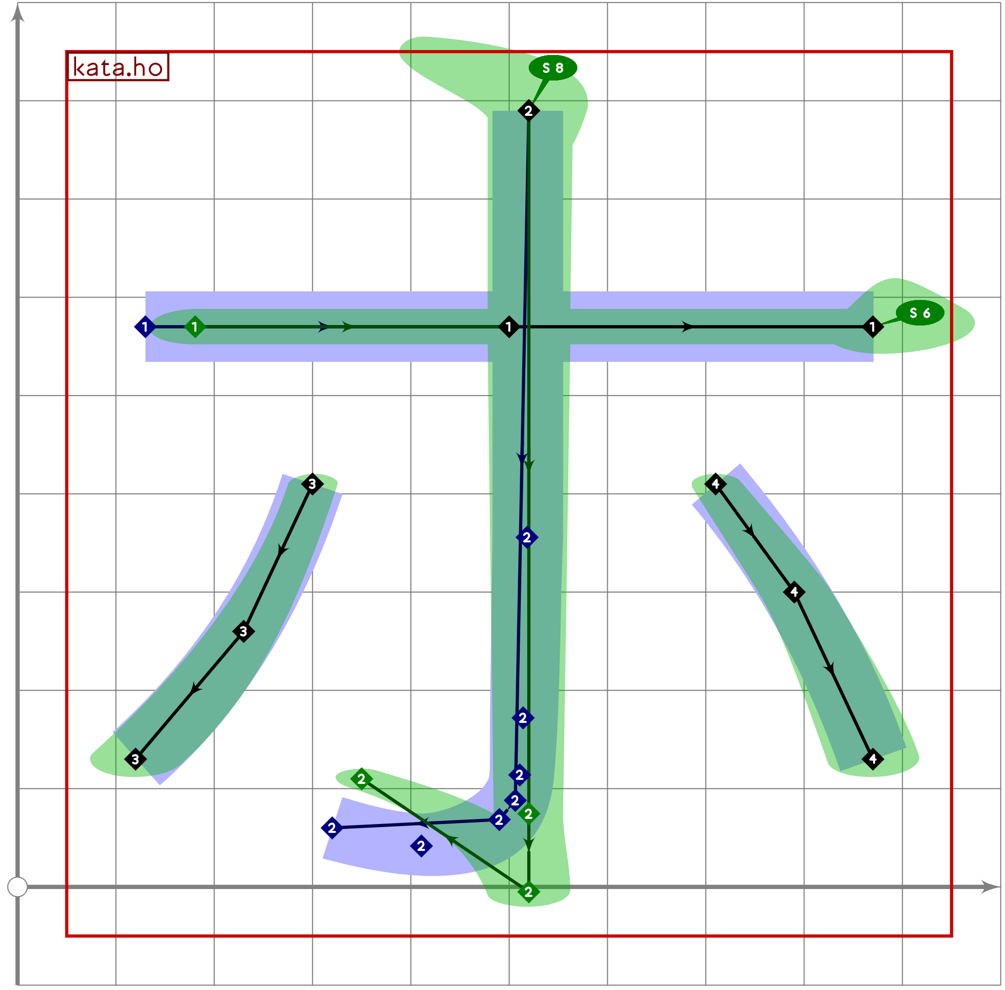
```

vardef  kata.fu  =
   push_pbox_toexpand(”kata.fu”);

   kata.fu_stroke((160,625),(790,635),(320,20));
   expand_pbox;
enddef;
```



```

vardef kata.he =
  push_pbox_toexpand("kata.he");

  push_stroke((120,320)--(440,570)--(880,230),
    (1.8,1.8)--(1.5,1.5)--(1.9,1.9));
  set_botip(0,1,1);
  set_boserif(0,0,5);
  expand_pbox;
enddef;

vardef kata.ho_centre(expr pta,ptb) =
  push_stroke(begingroup
    numeric x[],y[];
    path mycirc,ripx,ripy;
    mycirc:=fullcircle scaled 100 shifted ptb;
    z1=(pta-ptb) intersectionpoint mycirc;
    z2=ptb+(-200,40);
    z3=0.85[z2,z1];
    ripx:=pta{down}..tension 1.6..z3..{curl 0}z2;
    ripx:=pta{down}...(point 0.95 of ripx)..z3..{curl 0}z2;
```

U+30DB
tsuku.uni30DB

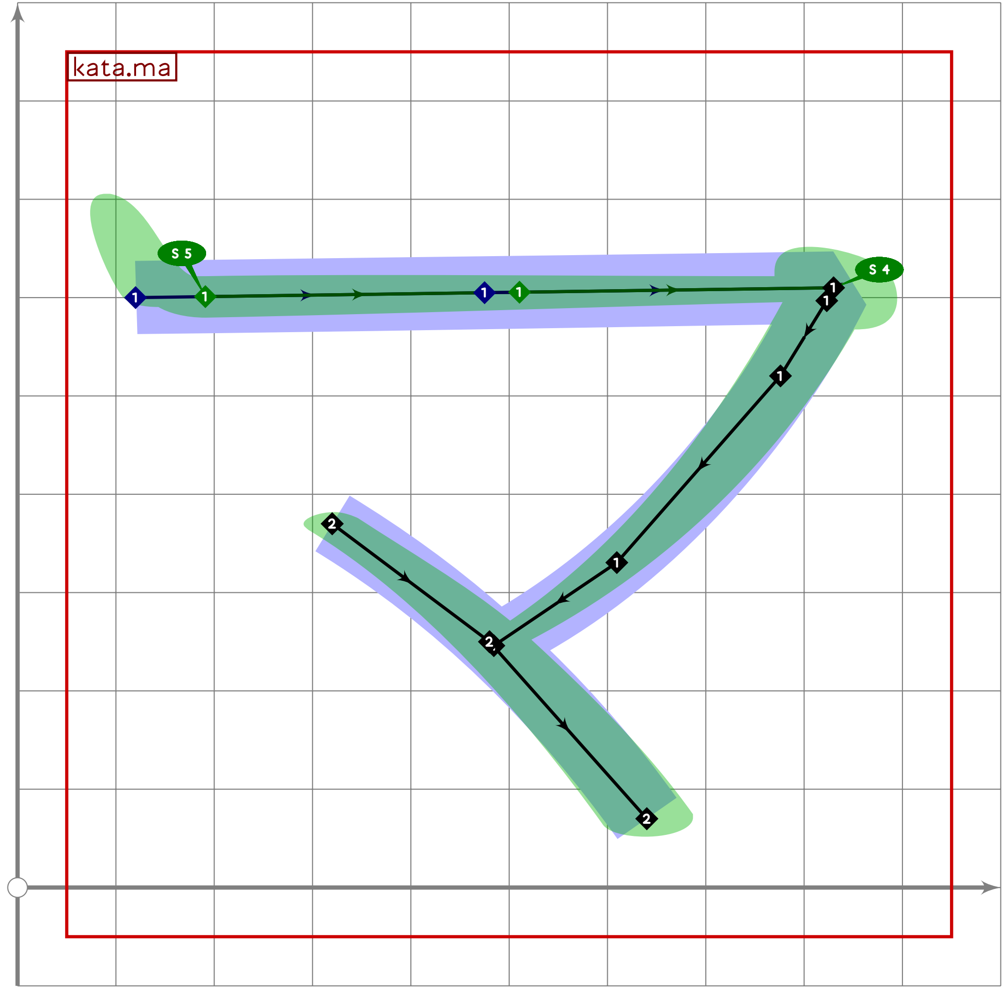
```
    z4=1.5[z1,ptb];
    z5=ptb+(-170,90);
    ripy:=pta--z4{z3-z4}..{curl 0}z5;
    ripy:=pta--(point 0.90 of ripy)--z4{z3-z4}..{curl 0}z5;;
    interpath(mincho,ripx,ripy)
  endgroup,
    (1.7,1.7)--(1.5,1.5)--(1.6,1.6)--(0.9,1.6));
  set_botip(0,2,0);
  set_boserif(0,0,8);
enddef;
```



```
vardef kata.ho =
  push_pbox_toexpand("kata.ho");

  push_stroke((130,570)--(500,570)--(870,570),
    (0.68,2.92)--(1.8,1.8)--(1.9,1.9));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  kata.ho_centre((520,790),(520,20));

  push_stroke((300,410)..(230,260)..(120,130),
    (1.2,1.2)--(1.5,1.5)--(1.8,1.8));

  push_stroke((710,410)..(790,300)..(870,130),
    (1.2,1.2)--(1.5,1.5)--(1.8,1.8));
  expand_pbox;
enddef;

```

Katakana Mamimumemo

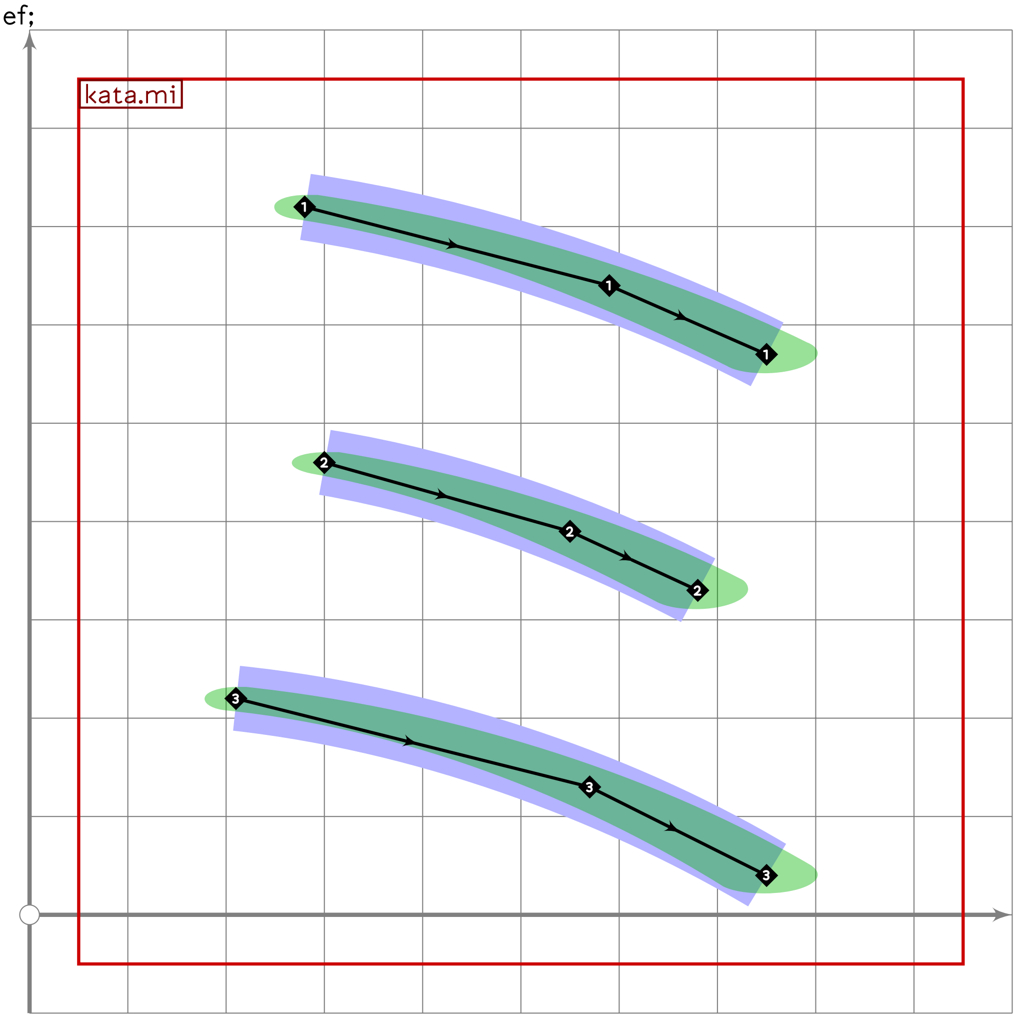
```
%%%%%%%%%% KATAKANA MAMIMUMEMO
```



```

vardef kata.ma =
  push_pbox_toexpand("kata.ma");

  kata.fu_stroke((120,600),(830,610),(200,180));

  push_stroke((320,370)..(480,250)..(640,70),
    (1.3,1.3)--(1.6,1.6)--(1.8,1.8));
  replace_strokep(-1)(subpath (0,xpart (oldp intersectiontimes
    get_strokep(0))) of oldp);
```

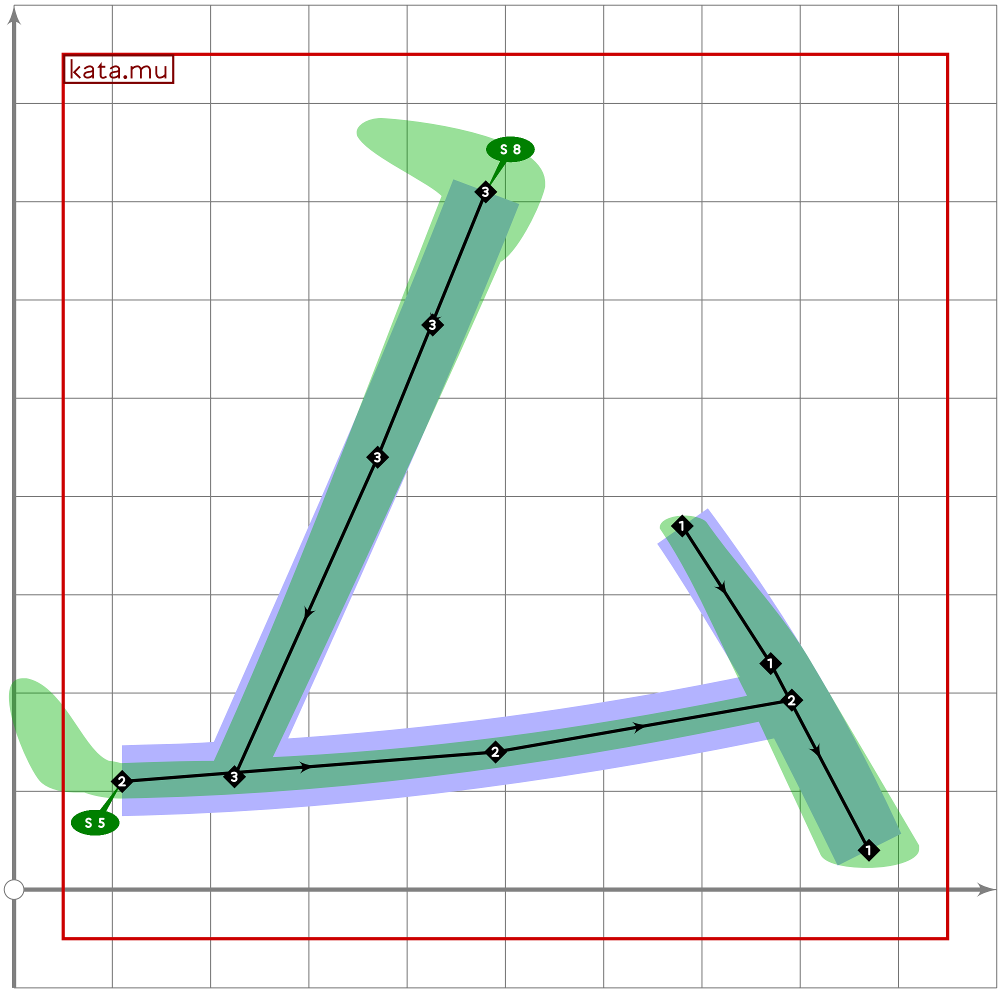
U+30DF
tsuku.uni30DF

```
  expand_pbox;
enddef;
```



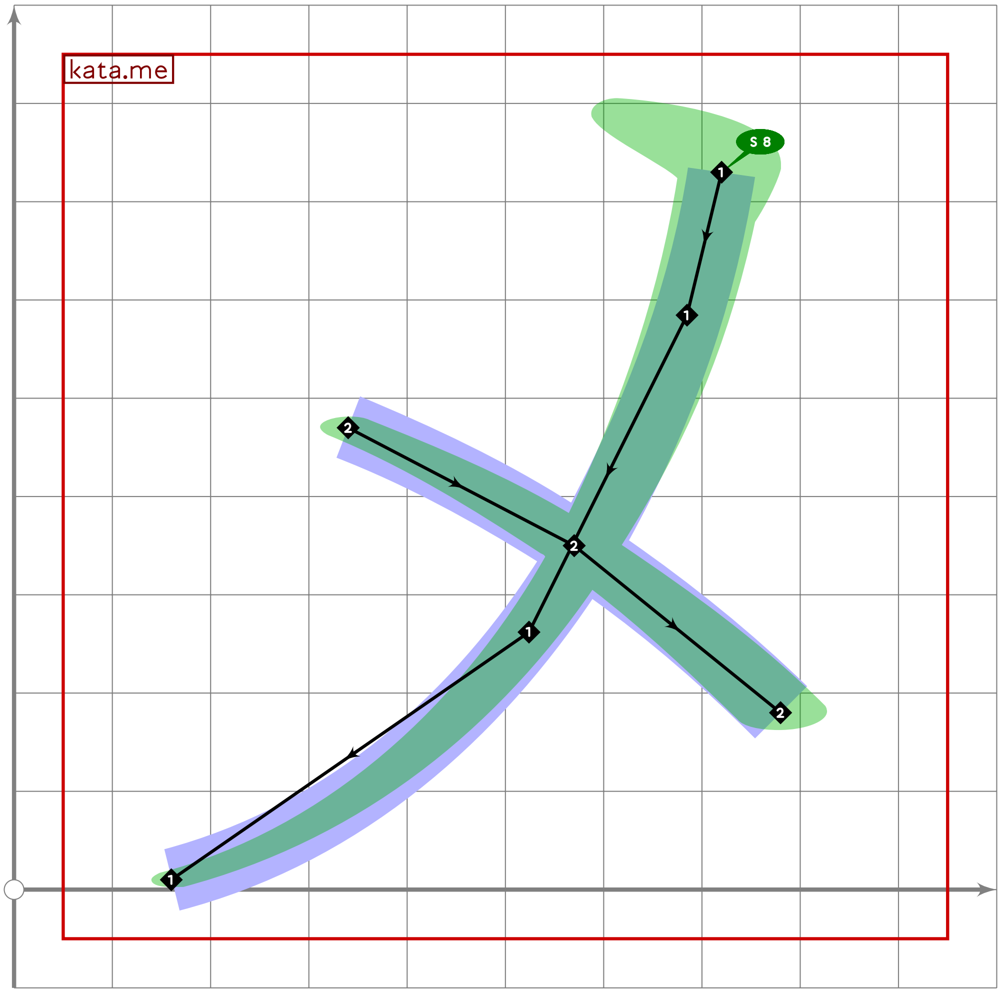
```

vardef  kata.mi =
  push_pbox_toexpand("kata.mi");

  push_stroke((280,720)..(590,640)..(750,570),
    (1.4,1.4)--(1.7,1.7)--(1.9,1.9));

  push_stroke((300,460)..(550,390)..(680,330),
    (1.4,1.4)--(1.7,1.7)--(1.9,1.9));

  push_stroke((210,220)..(570,130)..(750,40),
    (1.4,1.4)--(1.7,1.7)--(1.9,1.9));
  expand_pbox;
enddef;
```

U+30E0
tsuku.uni30E0



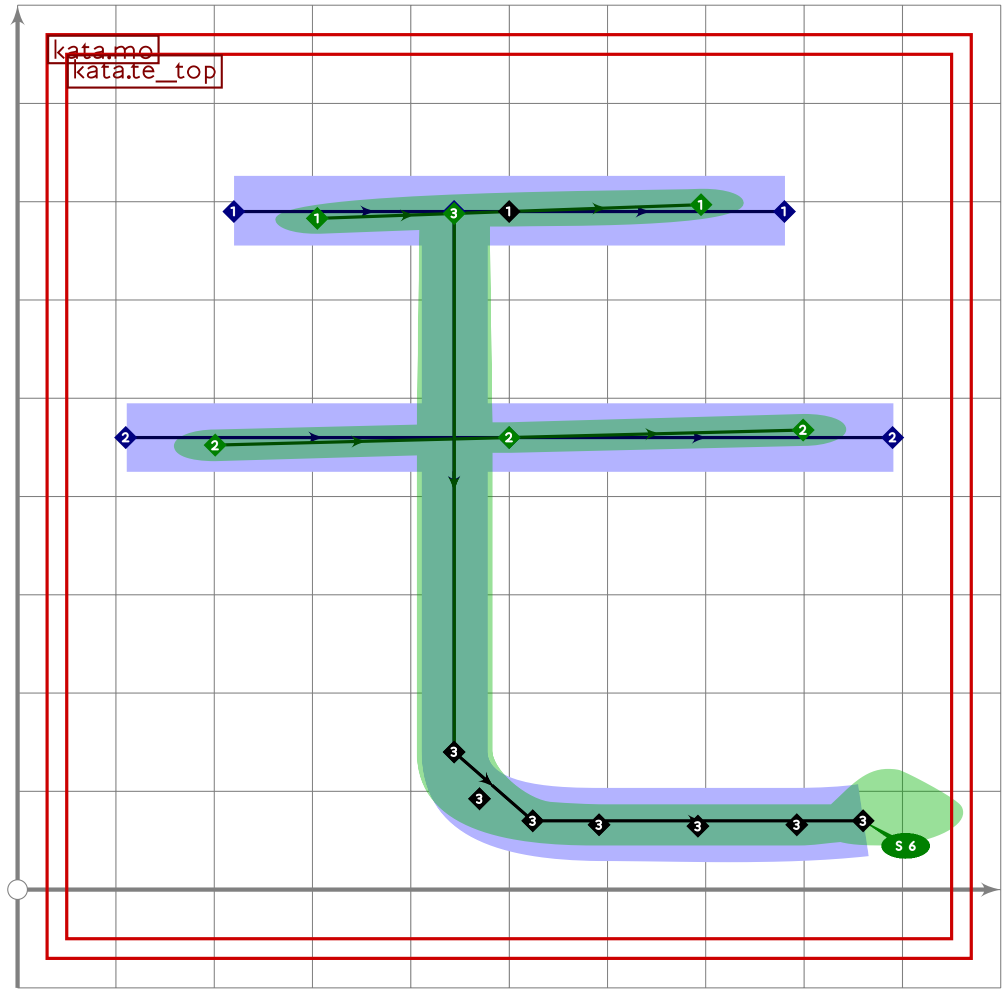
```
vardef kata.mu =
  push_pbox_toexpand(”kata.mu”);

  push_stroke((680,370)..(770,230)..(870,40),
    (1.2,1.2)–(1.6,1.6)–(1.9,1.9));

  push_stroke((110,110)..(490,140)..(point 1.2 of get_strokep(0)),
    (1.8,1.8)–(1.6,1.6)–(1.4,1.4));
  set_boserif(0,0,5);

  push_stroke((480,710)..(370,440)..(point 0.3 of get_strokep(0)),
    (1.7,1.7)–(1.5,1.5)–(1.3,1.3));
  set_boserif(0,0,8);
  expand_pbox;
enddef;
```

U+30E1
tsuku.uni30E1



```

vardef  kata.me =
   push_pbox_toexpand(”kata.me”);

   kata.no_stroke((720,730),(160,10));
   set_boserif(0,0,8);

   push_stroke((340,470)..(570,350)..(780,180),
      (1.3,1.3)–(1.6,1.6)–(1.8,1.8));
   expand_pbox;
enddef;
```

U+30E2
tsuku.uni30E2

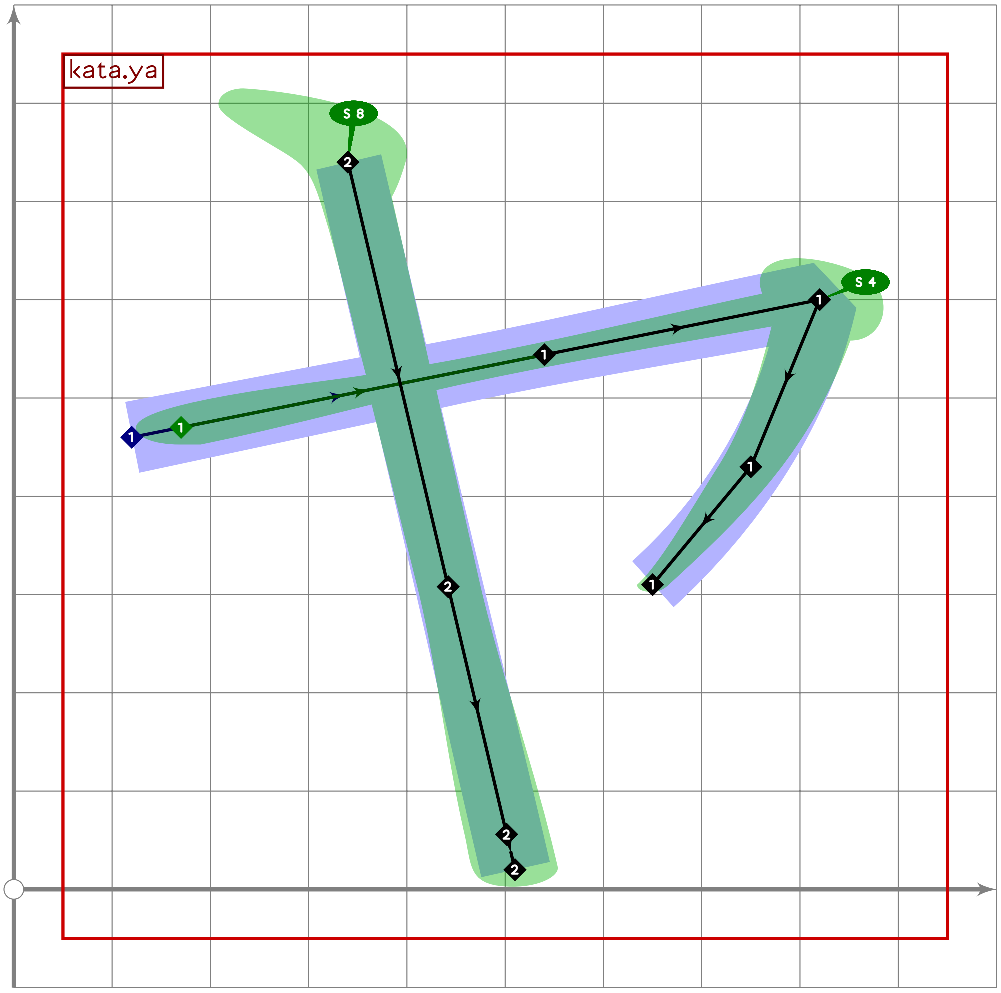


```

vardef kata.mo =
  push_pbox_toexpand("kata.mo");

  kata.te_top;

  push_stroke((point 0.8 of get_strokep(-1))--
      (xpart point 0.8 of get_strokep(-1),140){dir 274}..
      (80+xpart point 0.8 of get_strokep(-1),70)..tension 2.1..(860,70),
    (1.5,1.5)--(1.6,1.6)--(2,2)--(1.9,1.9));
  set_boserif(0,3,6);
  expand_pbox;
enddef;

```

Katakana Yayuyo

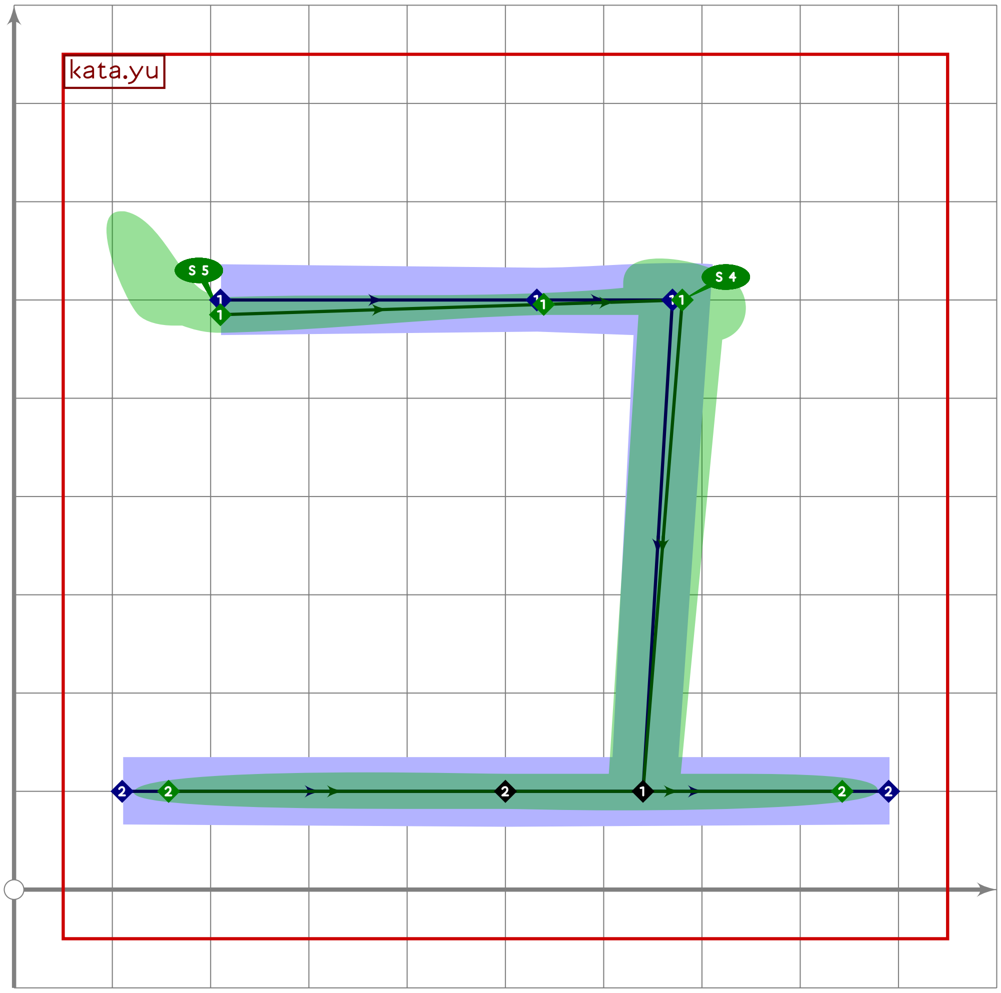
```
%%%%%%%%%% KATAKANA YAYUYO
```

U+30E4
tsuku.uni30E4



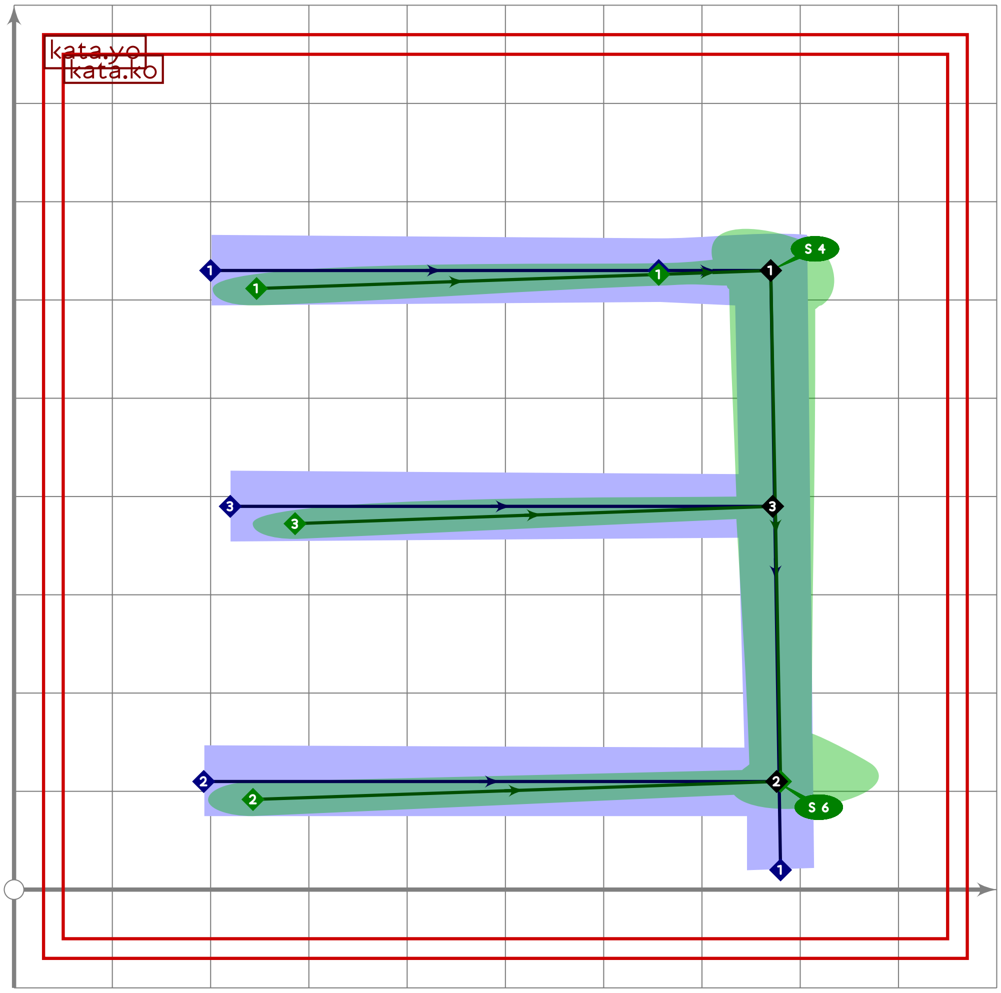
```
vardef  kata.ya =
  push_pbox_toexpand("kata.ya");

  push_stroke((120,460)−(820,600)..(750,430)..(650,310),
    (0.77,2.9)−(1.3,1.3)−(1.7,1.7)−(1.4,1.4)−(1,1));
  replace_strokep(0)(insert_nodes(oldp)(0.6));
  set_botip(0,2,0);
  set_boserif(0,0,5);
  set_boserif(0,2,4);

  push_stroke((340,740)−(510,20),
    (1.5,1.5)−(1.4,1.4)−(1.7,1.7)−(1.7,1.7));
  replace_strokep(0)(insert_nodes(oldp)(0.6,0.95));
  set_boserif(0,0,8);
  expand_pbox;
enddef;
```

U+30E6
tsuku.uni30E6



```
vardef  kata.yu =
  push_pbox_toexpand("kata.yu");

  push_stroke((210,600-15*mincho)--(670+10*mincho,600)--(640,100),
    (1.8,1.8)--(1.2,1.2)--(1.7,1.7)--(1.5,1.5));
  replace_strokep(0)(insert_nodes(oldp)(0.7));
  set_botip(0,2,1);
  set_boserif(0,0,5);
  set_boserif(0,2,4);

  push_stroke((110,100)--(500,100)--(890,100),
    (0.7,2.2)--(1.8,1.8)--(0.7,2.2));
  set_boserif(0,0,5);
  set_boserif(0,2,6);
  expand_pbox;
enddef;
```

U+30E8
tsuku.uni30E8

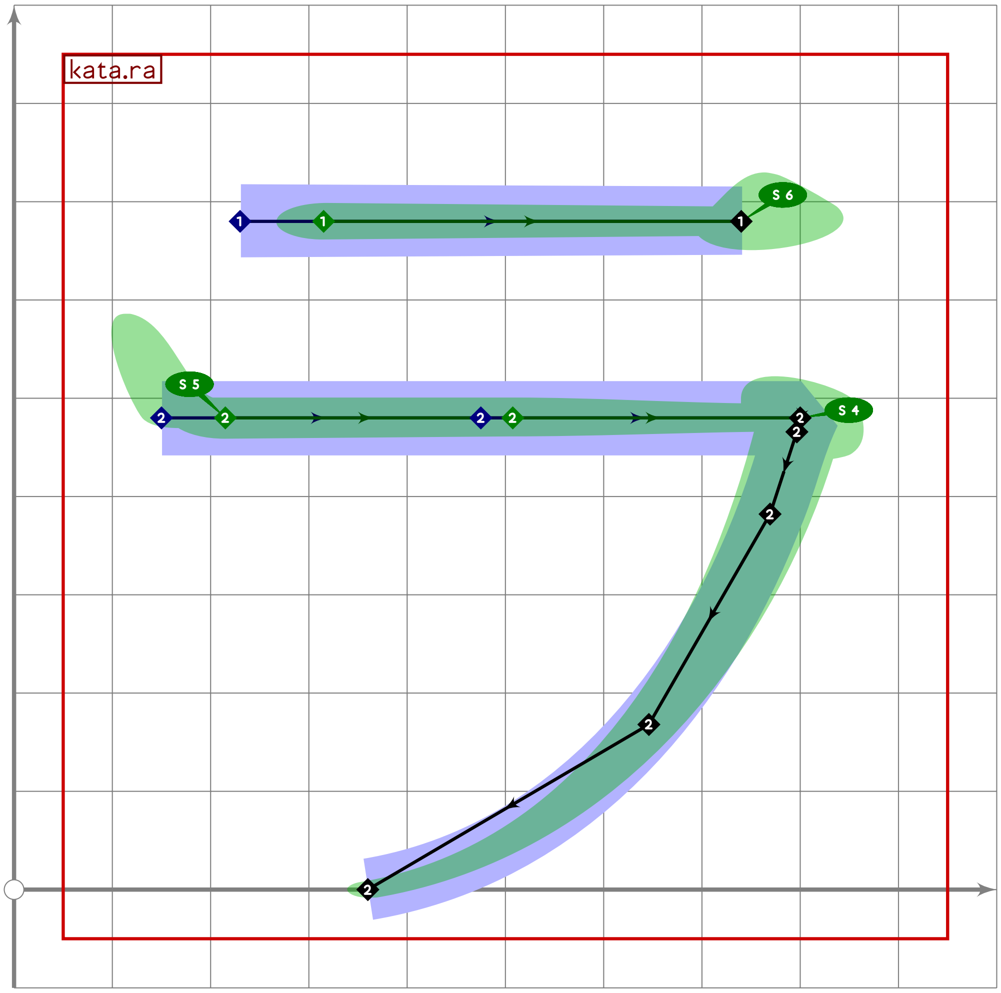


```

vardef kata.yo =
  push_pbox_toexpand("kata.yo");

  kata.ko;

  push_stroke((220,390-20*mincho)--(772,390),
    (0.77,2.7)--(1.3,1.3));
  set_boserif(0,0,5);
  expand_pbox;
enddef;

```

Katakana Rarirurero

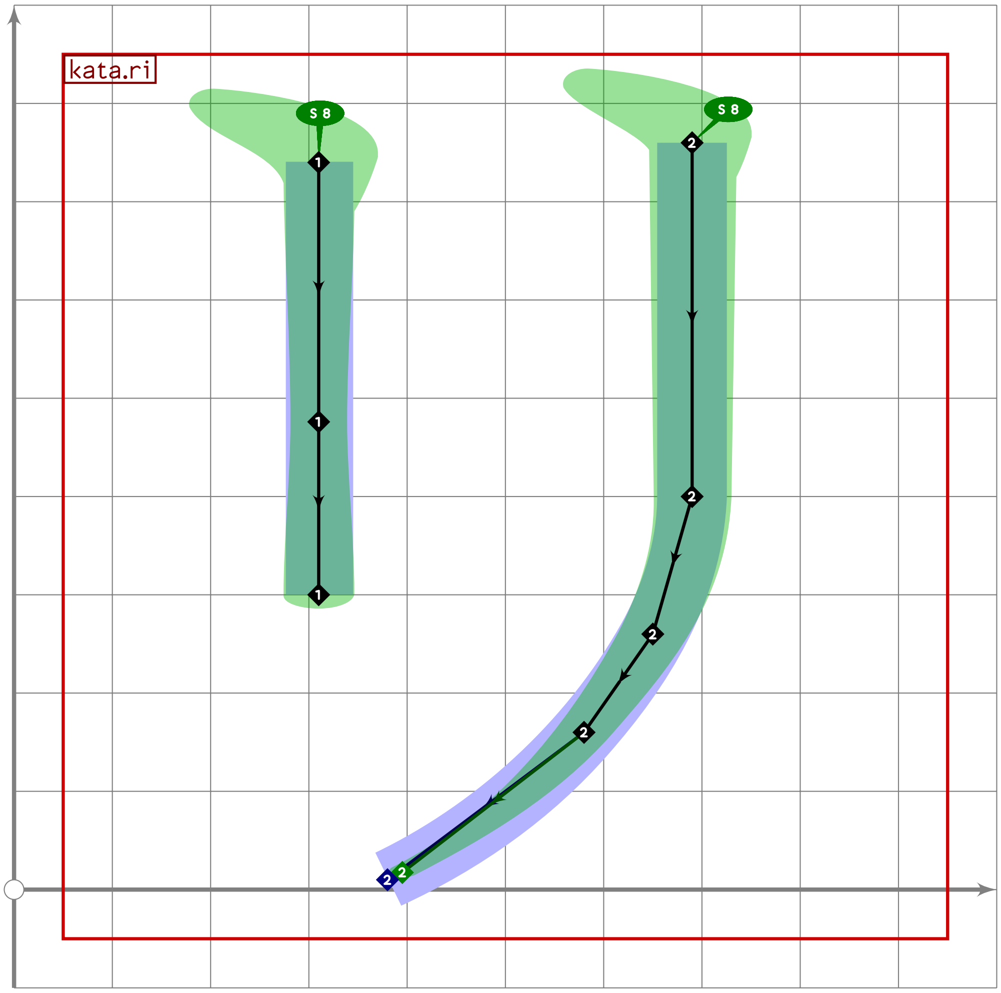
```
%%%%%%%%%%%% KATAKANA RARIRURERO
```

U+30E9
tsuku.uni30E9



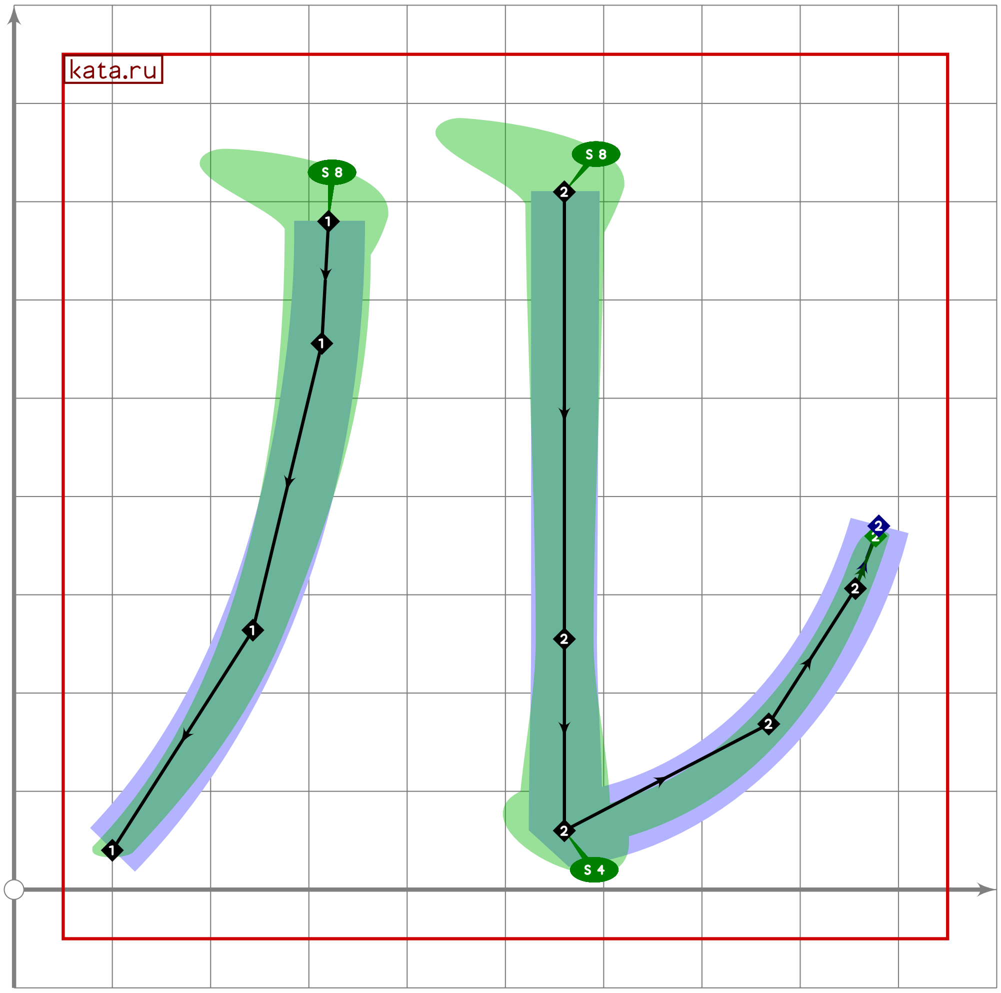
```

vardef  kata.ra =
   push_pbox_toexpand(”kata.ra”);

   push_stroke((230,680)–(740,680),
      (0.68,3.12)–(1.6,1.6));
   set_boserif(0,0,5);
   set_boserif(0,1,6);

   kata.fu_stroke((150,480),(800,480),(360,0));
   expand_pbox;
enddef;
```

U+30EA
tsuku.uni30EA



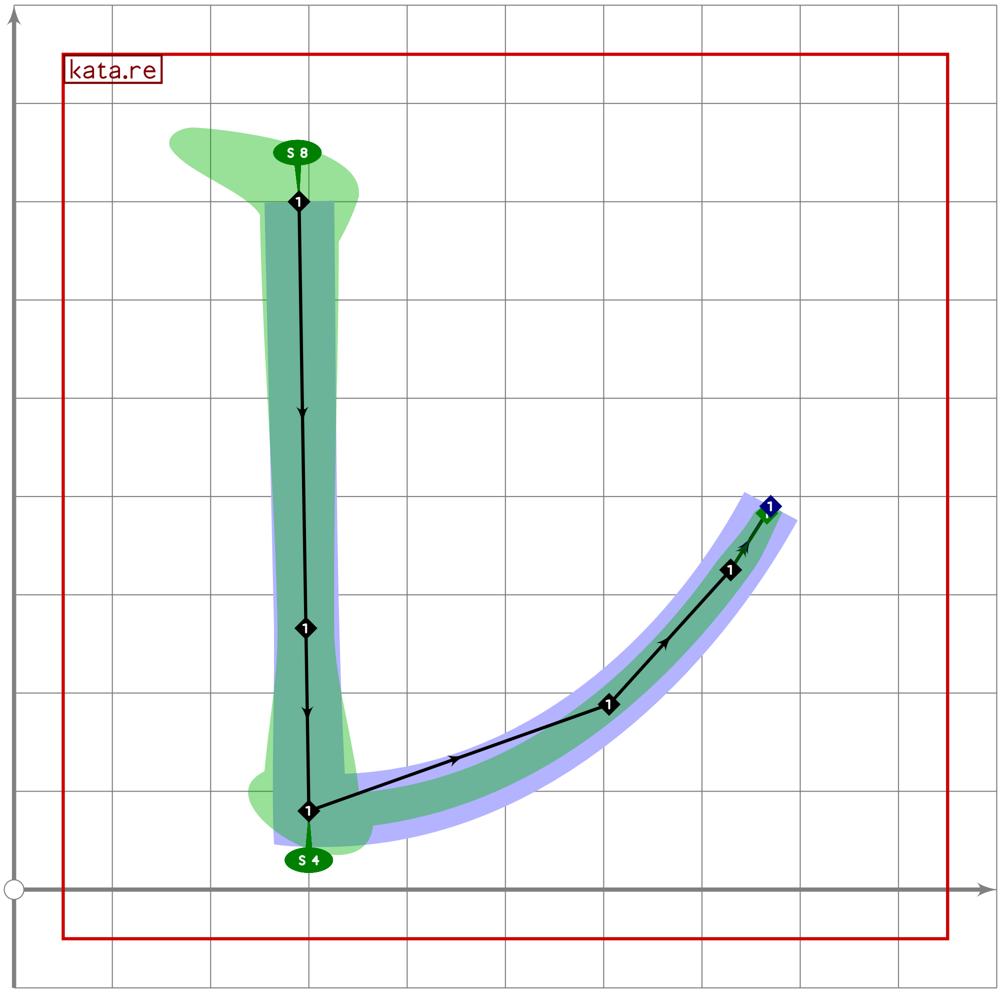
```

vardef kata.ri =
  push_pbox_toexpand(”kata.ri”);

  push_stroke((310,740)--(310,300),
    (1.5,1.5)--(1.3,1.3)--(1.5,1.5));
  replace_strokep(0)(insert_nodes(oldp)(0.6));
  set_boserif(0,0,8);

  push_stroke((690,760)--(690,400){dir 267}..(650,260)..(580,160)..(380,10),
    (1.7,1.7)--(1.6,1.6)--(1.5,1.5)--(1.3,1.3)--(0.8,1));
  set_boserif(0,0,8);
  expand_pbox;
enddef;
```

U+30EB
tsuku.uni30EB

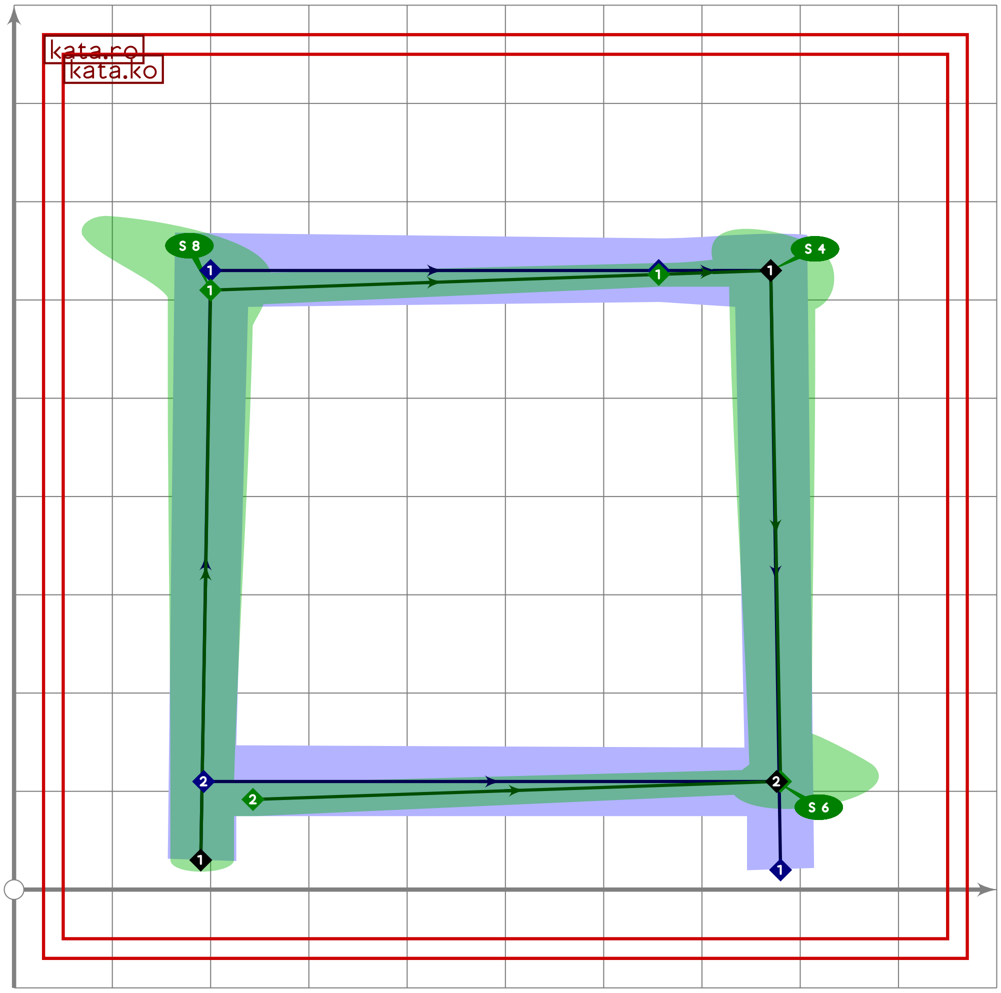


```
vardef kata.ru =
  push_pbox_toexpand("kata.ru");

  kata.no_stroke((320,680),(100,40));
  set_boserif(0,0,8);

  kata.no_stroke((880,370),(560,60));
  replace_strokep(0)(insert_nodes((560,710)--reverse oldp)(0.7));
  replace_strokeq(0)((1.6,1.6)--(1.3,1.3)--(1.8,1.8)--
      (1.2,1.2)--(1,1)--(0.8,1));
  set_botip(0,2,0);
  set_boserif(0,0,8);
  set_boserif(0,2,4);
  expand_pbox;
enddef;
```

U+30EC
tsuku.uni30EC



```

vardef kata.re =
  push_pbox_toexpand(”kata.re”);

  kata.no_stroke((770,390),(300,80));
  replace_strokep(0)(insert_nodes((290,700)--reverse oldp)(0.7));
  replace_strokeq(0)((1.6,1.6)--(1.3,1.3)--(1.8,1.8)--
      (1.2,1.2)--(1.1,1.1)--(0.8,1));
  set_botip(0,2,1);
  set_boserif(0,0,8);
  set_boserif(0,2,4);
  expand_pbox;
enddef;
```

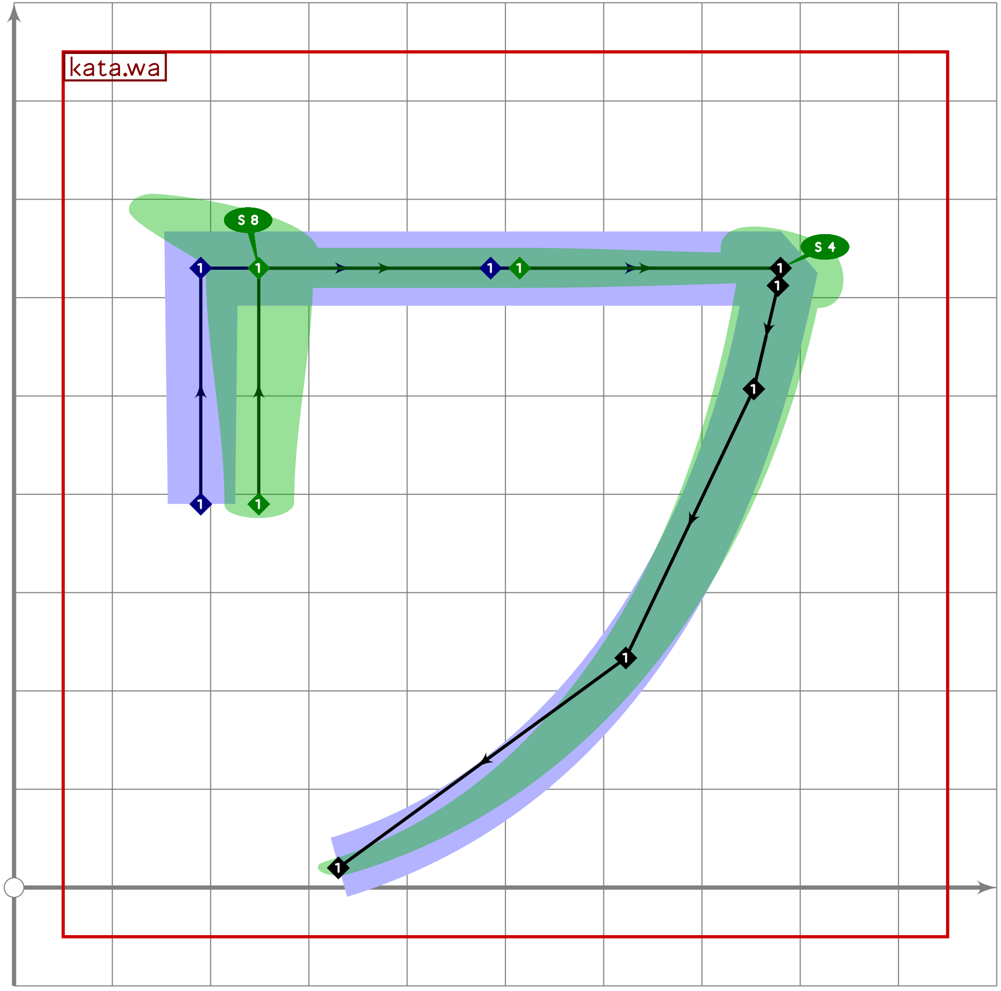


```
vardef kata.ro =
  push_pbox_toexpand("kata.ro");

  kata.ko;

  replace_strokep(-1)((190,30)--oldp);
  replace_strokeq(-1)((1.4,1.4)--(1.7,1.7)--(subpath (1,infinity) of oldq));
  set_botip(-1,1,1);
  set_botip(-1,2,whatever);
  set_botip(-1,3,1);
  set_boserif(-1,0,whatever);
  set_boserif(-1,1,8);
  set_boserif(-1,2,whatever);
  set_boserif(-1,3,4);
  set_boserif(0,0,whatever);
  expand_pbox;
enddef;
```

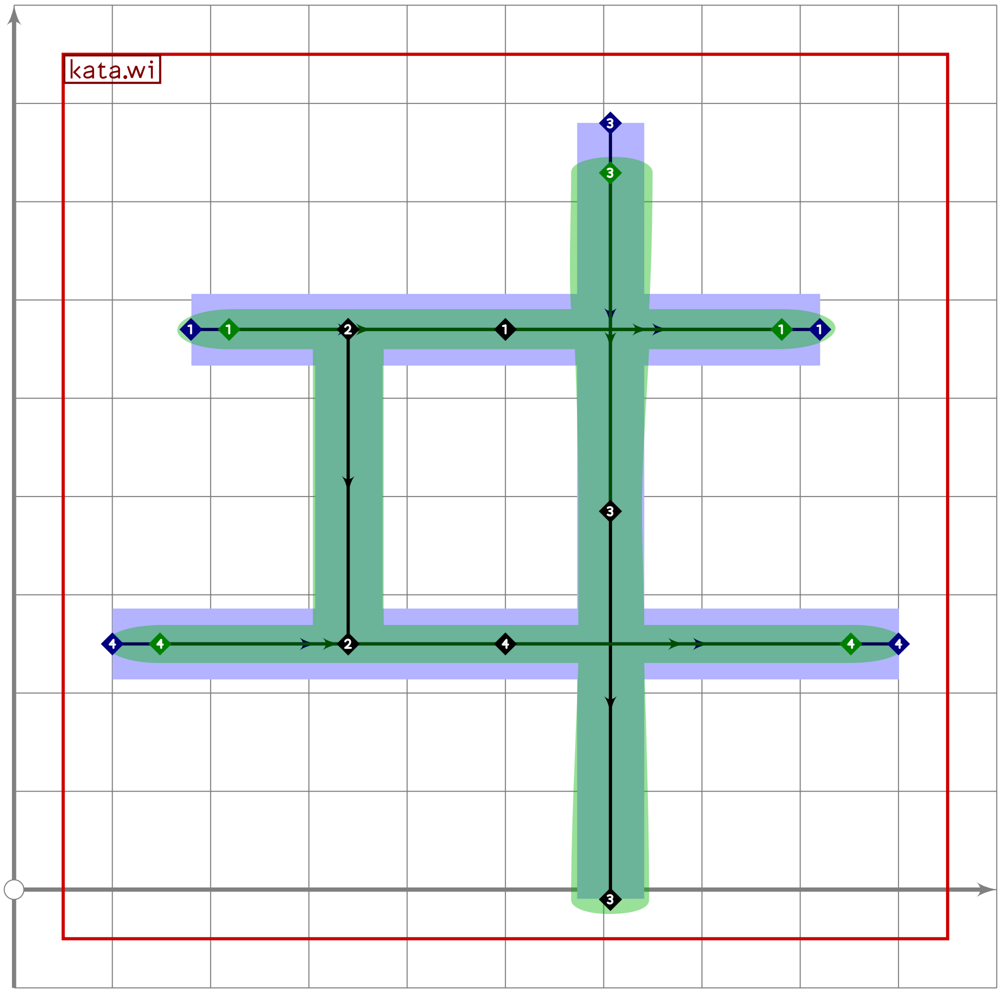
U+30EF
tsuku.uni30EF

Katakana Wawiwewo/N/Iteration

```
%%%%%%%%%% KATAKANA WAWIWEWO/N/ITERATION
```



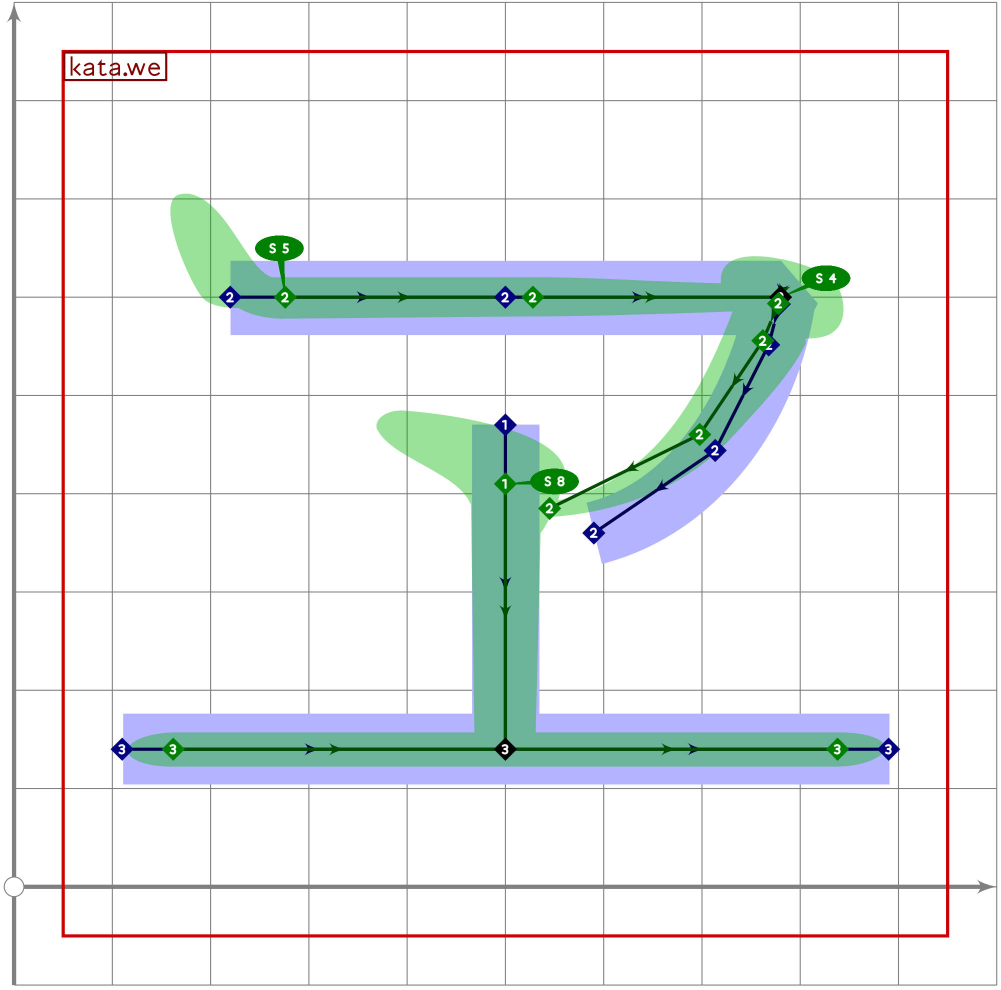
```

vardef kata.wa =
  push_pbox_toexpand("kata.wa");

  kata.fu_stroke((190,630),(780,630),(330,20));
  replace_strokep(0)((xpart point 0 of oldp,390)--oldp);
  replace_strokeq(0)((1.5,1.5)--oldq);
  set_botip(0,1,1);
  set_botip(0,3,0);
  set_boserif(0,0,whatever);
  set_boserif(0,1,8);
  set_boserif(0,2,whatever);
  set_boserif(0,3,4);
  expand_pbox;
enddef;
```



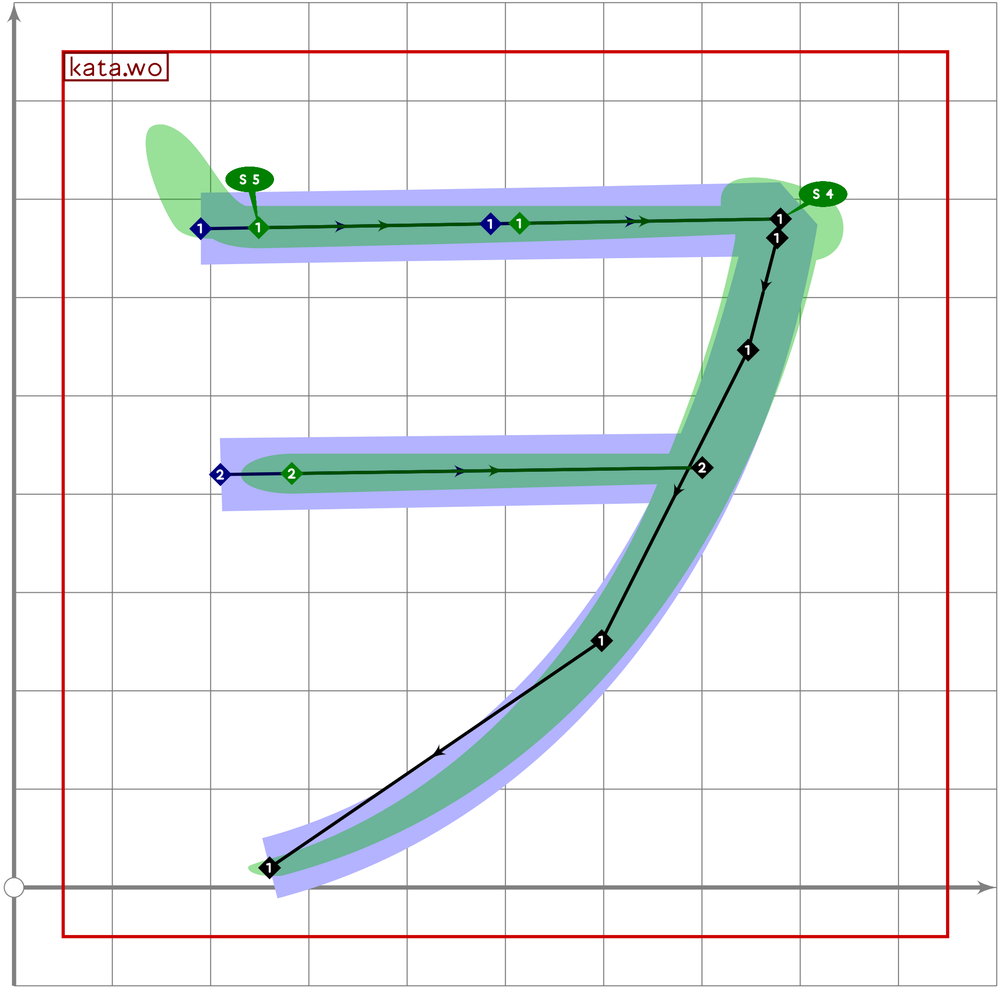
```

vardef kata.wi =
  push_pbox_toexpand(”kata.wi”);

  x1=100;
  x2=180;
  x3=0.25[x2,x5];
  x4=0.667[x2,x5];
  (x5+x2)/2=(x1+x6)/2=500;
  y1=-10;
  y2=250;
  y3=570;
  y4=780;
  push_stroke((x2,y3)–(x5,y3),
    (0.7,3.3)–(1.8,1.8)–(0.7,3.3));
  replace_strokep(0)(insert_nodes(oldp)(0.5));
  set_boserif(0,0,5);
  set_boserif(0,2,6);

  push_stroke((x3,y3)–(x3,y2),
    (1.5,1.5)–(1.5,1.5));
```

U+30F1
tsuku.uni30F1

```

  push_stroke((x4,y4)--(x4,0.5[y4,y1])--(x4,y1),
    (0.75,2.65)--(1.4,1.4)--(1.6,1.6));
  set_boserif(0,0,8);

  push_stroke((x1,y2)--(x6,y2),
    (0.7,3.3)--(1.8,1.8)--(0.7,3.3));
  replace_strokep(0)(insert_nodes(oldp)(0.5));
  set_boserif(0,0,5);
  set_boserif(0,2,6);
  expand_pbox;
enddef;
```

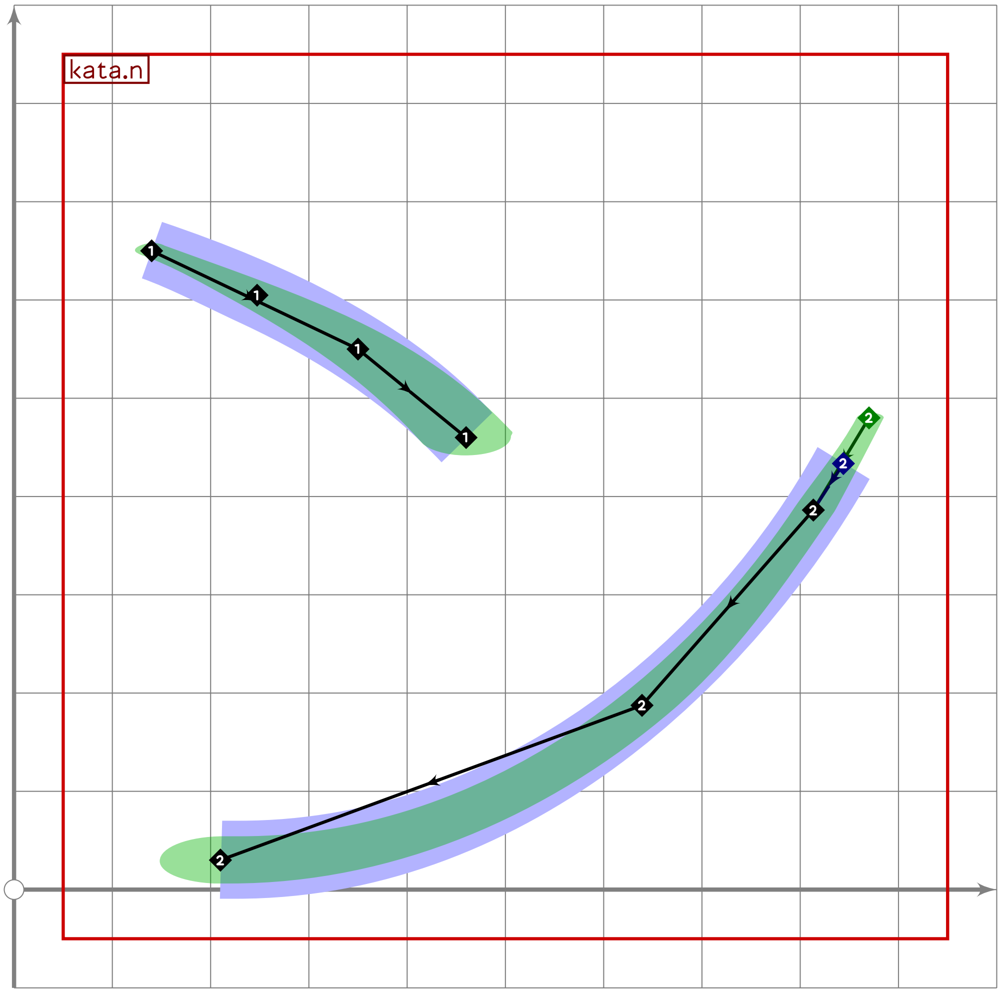


```

vardef kata.we =
  push_pbox_toexpand("kata.we");

  push_stroke((500,470-60*mincho)--(500,140),
    (1.5,1.5)--(1.4,1.4));
  set_boserif(0,0,8);

  kata.fu_stroke((220,600),(780,600),
    (0.5*mincho)[(590,360),point 0 of get_strokep(0)]);

  push_stroke((110,140)--(500,140)--(890,140),
    (0.7,2.9)--(1.7,1.7)--(0.7,2.9));
  set_boserif(0,0,5);
  set_boserif(0,2,6);
  expand_pbox;
enddef;
```



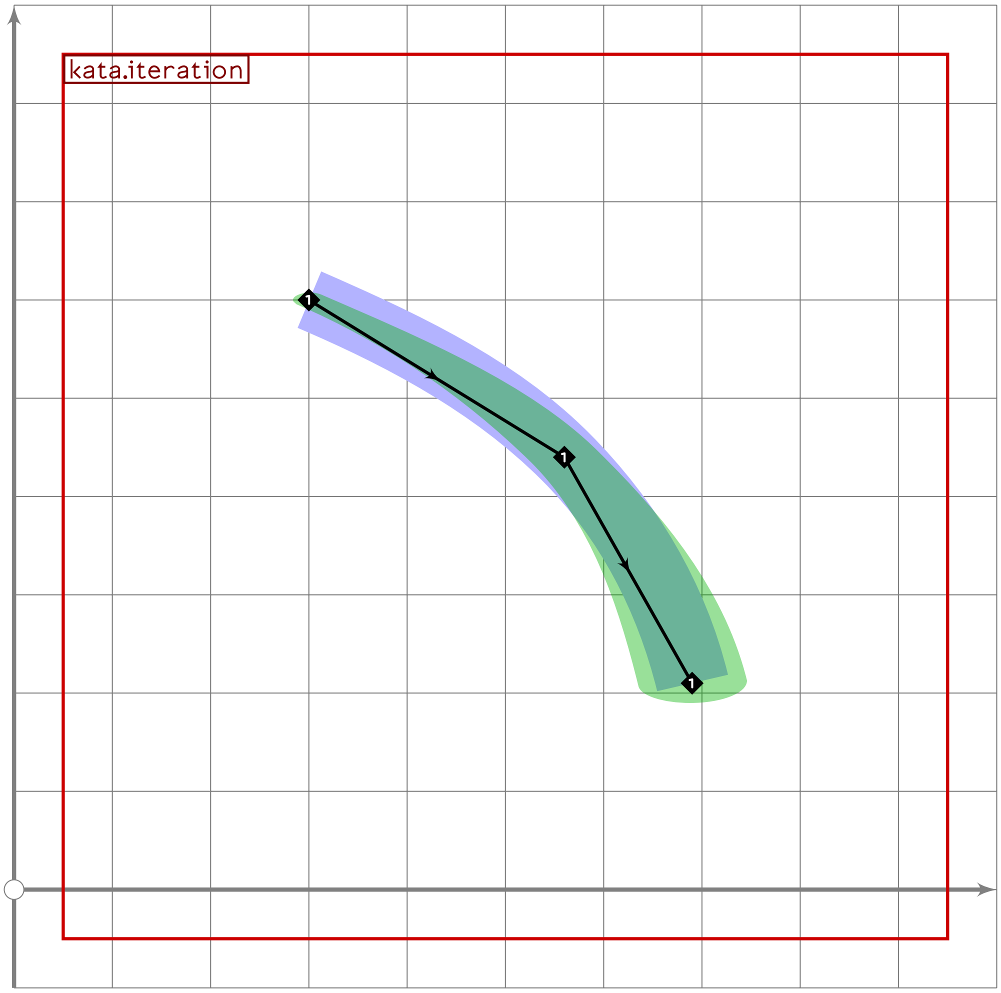
```

vardef kata.wo =
  push_pbox_toexpand("kata.wo");

  kata.fu_stroke((190,670),(780,680),(260,20));

  z1=get_strokep(0) intersectionpoint ((0,420)--(1000,430));
  push_stroke((210,420)--z1,
    (0.7,3.3)--(1.6,1.6));
  set_boserif(0,0,5);
  expand_pbox;
enddef;
```

U+30F3
tsuku.uni30F3



```
vardef  kata.n  =
  push_pbox_toexpand("kata.n");

  push_stroke((140,650)..tension  1.2..(350,550)..(460,460),
    (1,1)..(1.6,1.6)..(1.8,1.8));

  kata.no_stroke((870,480),(210,30));
  replace_strokeq(0)((0.9,0.9)--(1.1,1.1)--(1.4,1.4)--(2.2,2.2));
  set_boserif(0,4,5);
  expand_pbox;
enddef;
```

U+30FD
tsuku.uni30FD



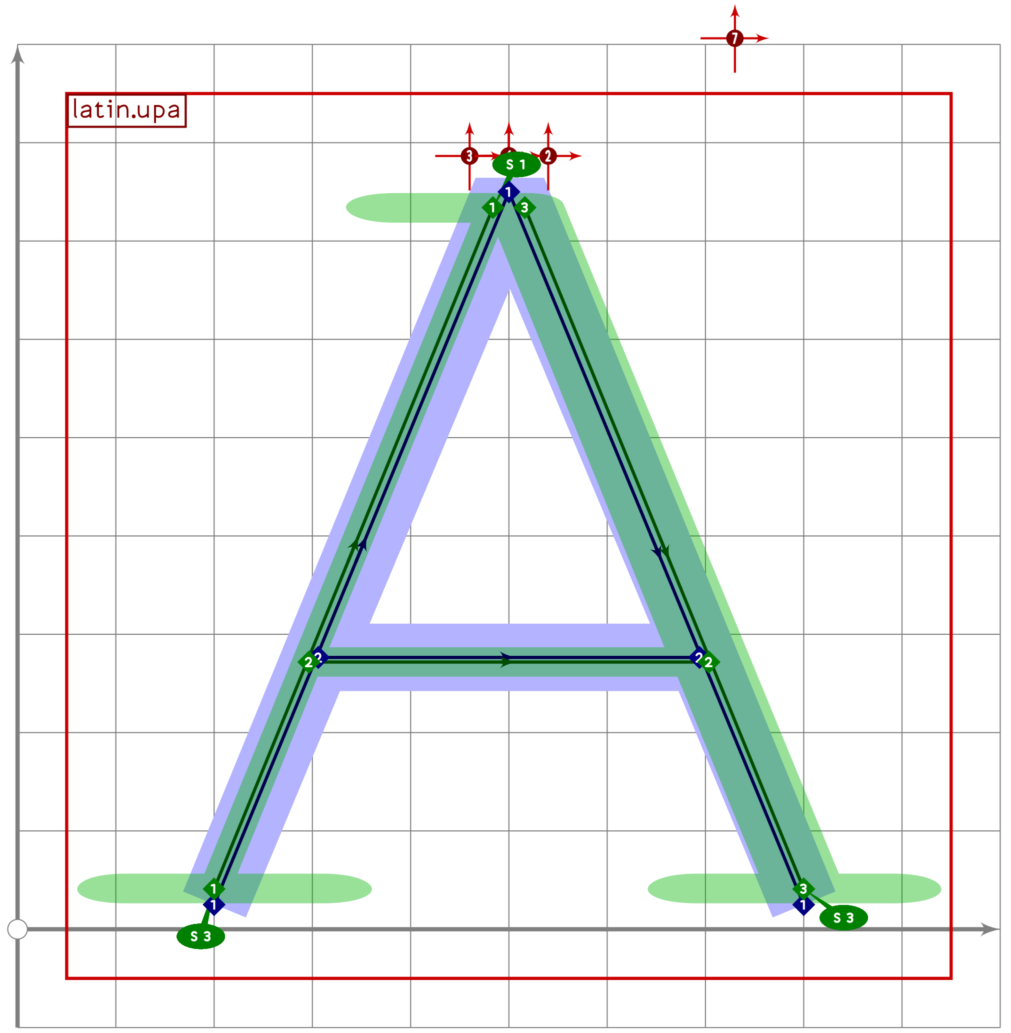
```
vardef kata.iteration =
  push_pbox_toexpand(”kata.iteration”);

  push_stroke((300,600){curl 0.2}..(560,440)..(690,210),
    (1,1)--(1.5,1.5)--(2,2));
  expand_pbox;
enddef;
```

KATA

U+FF21
tsuku.uniFF21

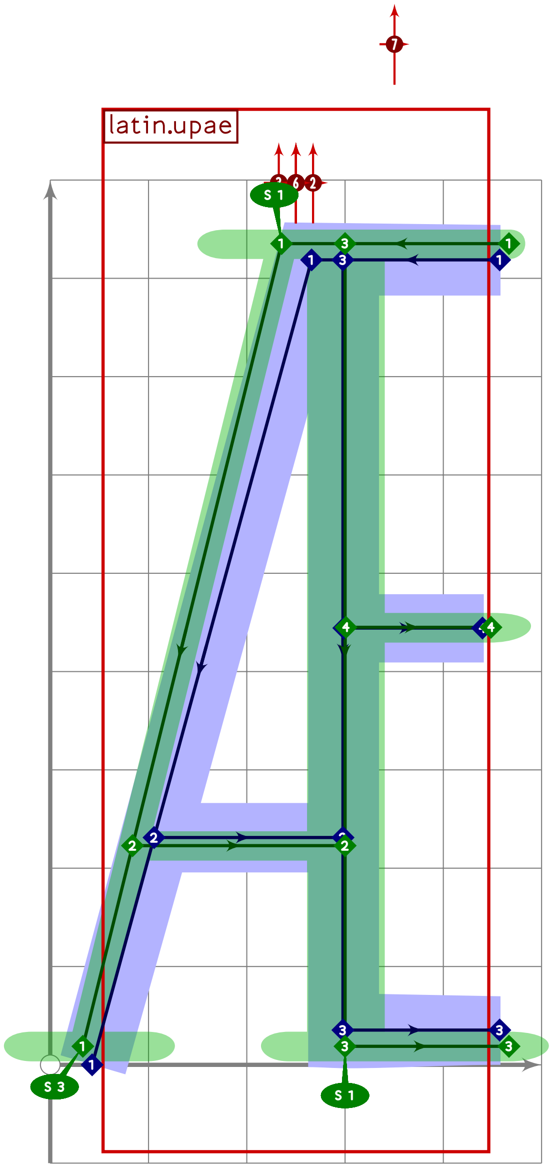
latin.mp

```
%
% Latin and related letters for Tsukurimashou
% Copyright (C) 2011, 2012, 2013 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(latin);

```



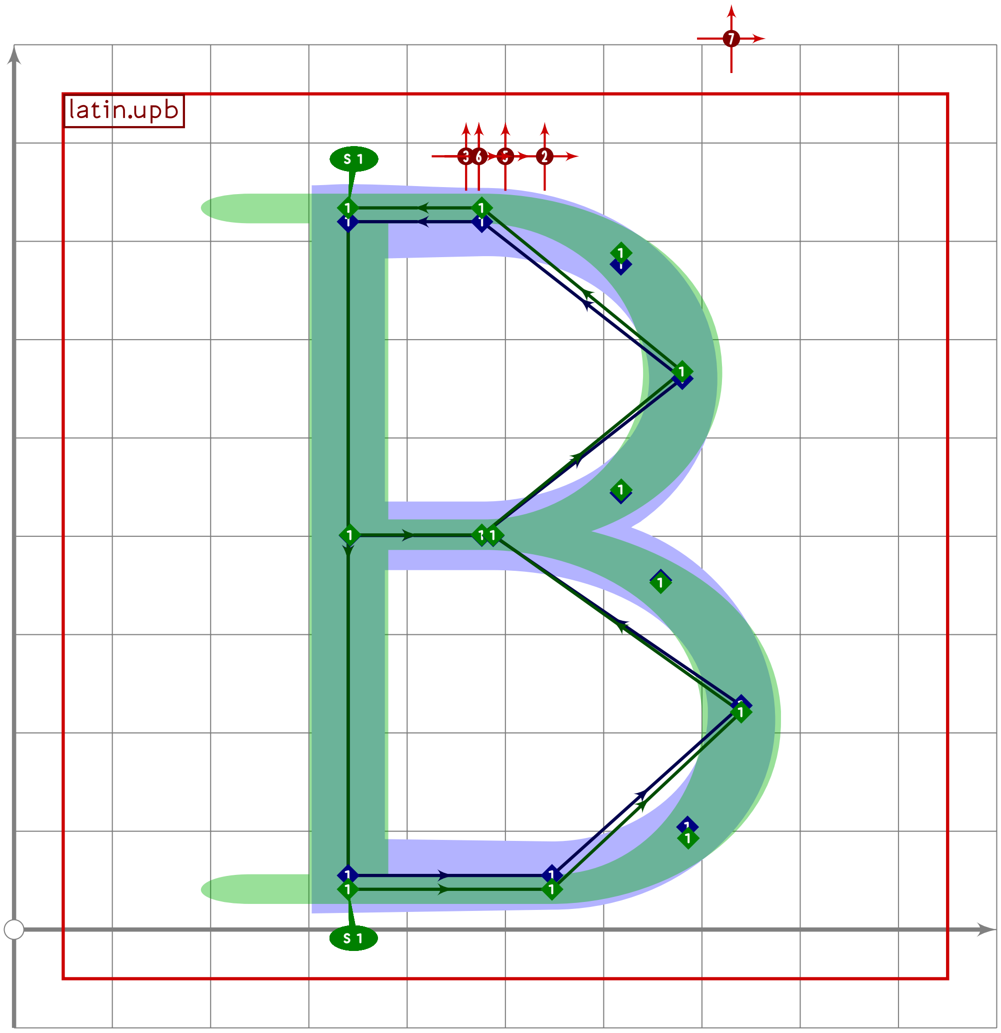
```

vardef latin.upa =
  push_pbox_toexpand("latin.upa");
  z1=(200,latin_wide_low_v);
  z2=(500,latin_wide_high_v);
  z3=(800,latin_wide_low_v);

  if do_alternation:
    z4=whatever[z1,(z2+alternate_adjust*left/2)]+(2,0);
    z5=whatever[(z2+alternate_adjust*right/2),z3]-(2,0);
    y4=y5=vmetric(0.333);

    push_stroke(z1--(z2+alternate_adjust*left/2),(1.6,1.6)--(1.6,1.6));
    set_boalternate(0);
    set_botip(0,1,0);
    set_boserif(0,0,3);
    set_boserif(0,1,1);

    push_stroke(z4--z5,(1.6,1.6)--(1.6,1.6));
    set_boalternate(0);

    push_stroke((z2+alternate_adjust*right/2)--z3,(1.6,1.6)--(1.6,1.6));
    set_boserif(0,1,3);
  else:
    z4=whatever[z1,z2]+(2,0);
    z5=whatever[z2,z3]-(2,0);
    y4=y5=vmetric(0.333);

    push_stroke(z1--z2--z3,(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
    set_botip(0,1,0);
    set_boserif(0,0,3);
    set_boserif(0,1,1);
    set_boserif(0,2,3);

    push_stroke(z4--z5,(1.6,1.6)--(1.6,1.6));
  fi;

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  expand_pbox;
enddef;
```

U+00C6
tsuku.AE

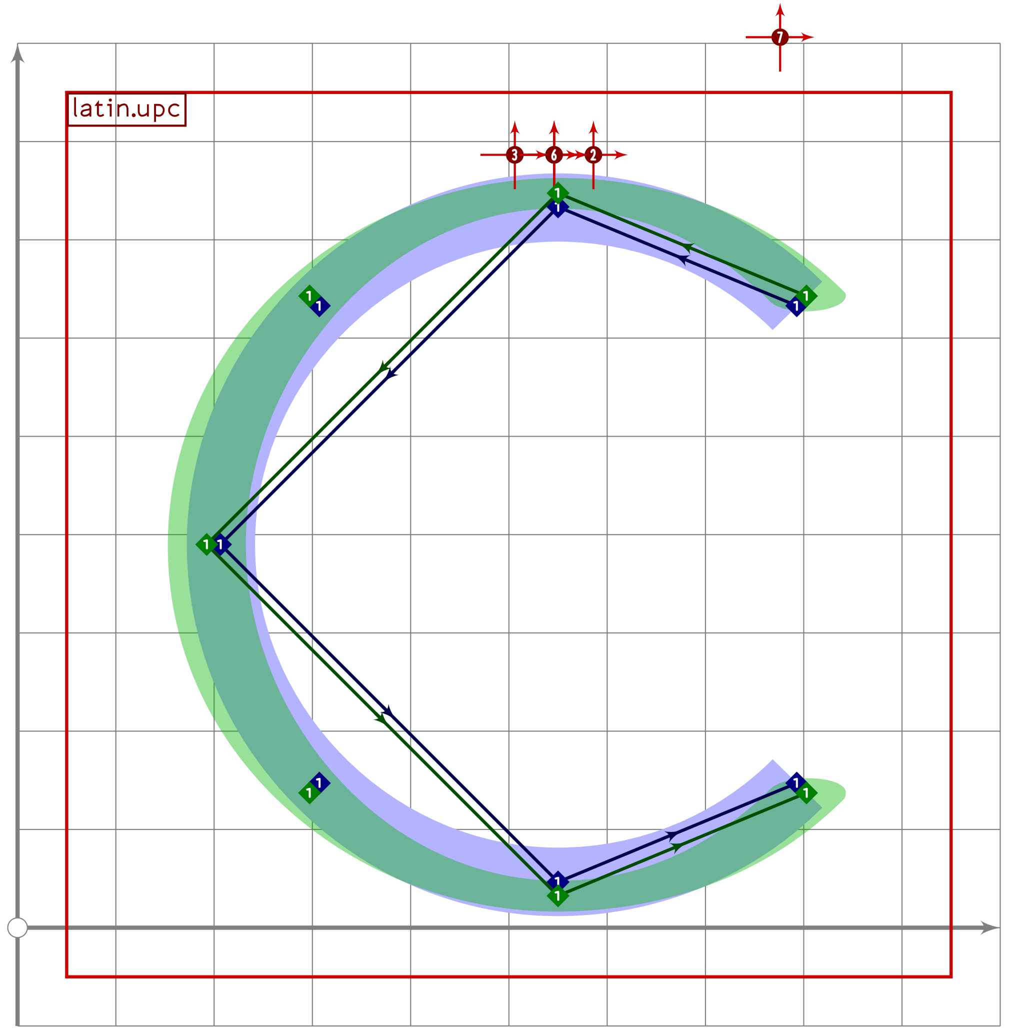


```
vardef latin.upae =
  push_pbox_toexpand(”latin.upae”);
  y1=y2=latin_wide_high_h;
  y3=y4=latin_wide_low_h;
  y5=y6=vmetric(0.522);

  (x1+x7)/2=500;
  x2=x3;
  x1=x4;
  x5=x2+2;
  x6=0.89[x2,x1];
  (x1-x2)=(y2-y3)*0.55;

  y7=latin_wide_low_v;
  x7=(-1.6)[x2,x1];
  z10=(-0.2)[z2,z1]+2.2*alternate_adjust*left;
  z8=whatever[z7,z10];
  z9=whatever[z2,z3];
  y8=y9=vmetric(0.250);

  push_stroke(z1--z10--z7,(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,1,1);
  set_boserif(0,1,1);
  set_boserif(0,2,3);
  set_boalternate(0);

  push_stroke(z8--z9,(1.6,1.6)--(1.6,1.6));
  set_boalternate(0);

  push_stroke(z2--z3--z4,(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,1,1);
  set_boserif(0,1,1);

  push_stroke(z5--z6,(1.6,1.6)--(1.6,1.6));

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  expand_pbox;
enddef;
```

U+FF22
tsuku.uniFF22



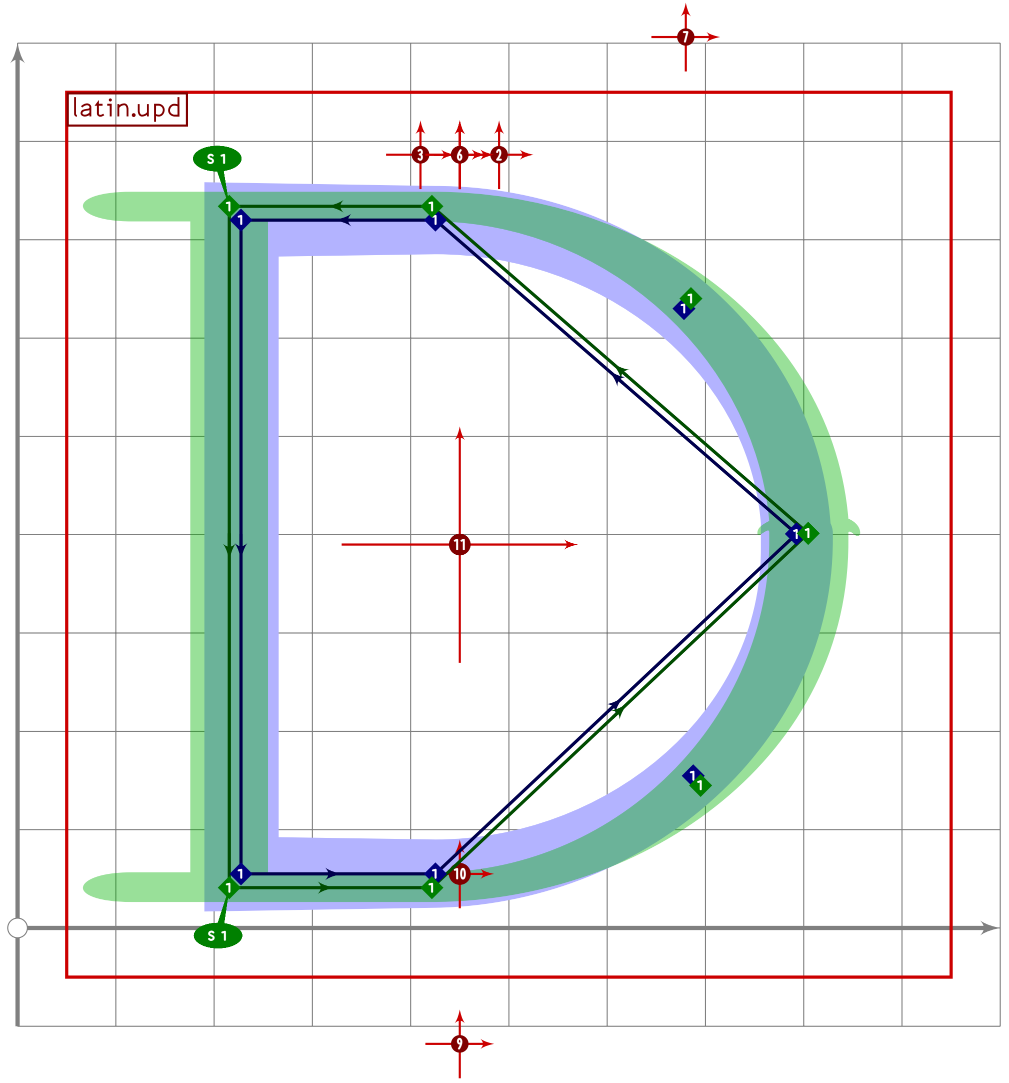
```

vardef latin.upb =
  push_pbox_toexpand(”latin.upb”);
  latin.upp_base(340);

  x6=x5;
  x7=0.61[x5,x3];
  x8=x5+400;

  y6=y7=latin_wide_low_h;
  y8=0.5[y6,y1];

  z9=z2;

  replace_strokep(0)(oldp–z6–z7{right}..z8..{left}z9);
  replace_strokep(0)(subpath (0,7.97) of oldp);
  replace_strokeq(0)(oldq–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6));

  set_botip(0,4,1);
  set_botip(0,5,1);
  set_boserif(0,4,1);
  set_boserif(0,5,1);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  tsu_accent.shift_anchors((ai=anc_ring) or (ai=anc_upper))((-27,0));
  expand_pbox;
enddef;
```



```

vardef latin.upc =
  push_pbox_toexpand("latin.upc");
  push_stroke(
    (subpath (0.5,3.5) of ((1,0)..(0,1)..(-1,0)..(0,-1)..cycle))
        scaled ((latin_wide_high_r-latin_wide_low_r)/2)
        shifted (centre_pt+(50,0)),
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
```

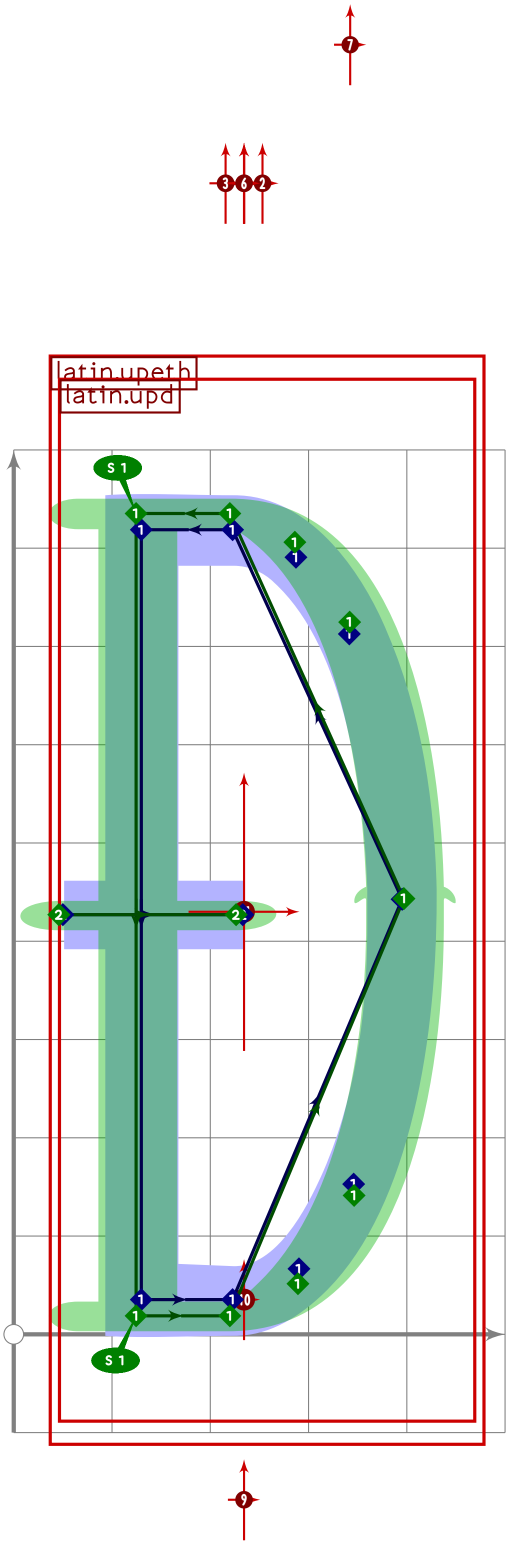
U+FF24
tsuku.uniFF24

```
      ((0,0) transformed accent_default[anc_upper])+(46,0));
  expand_pbox;
enddef;
```

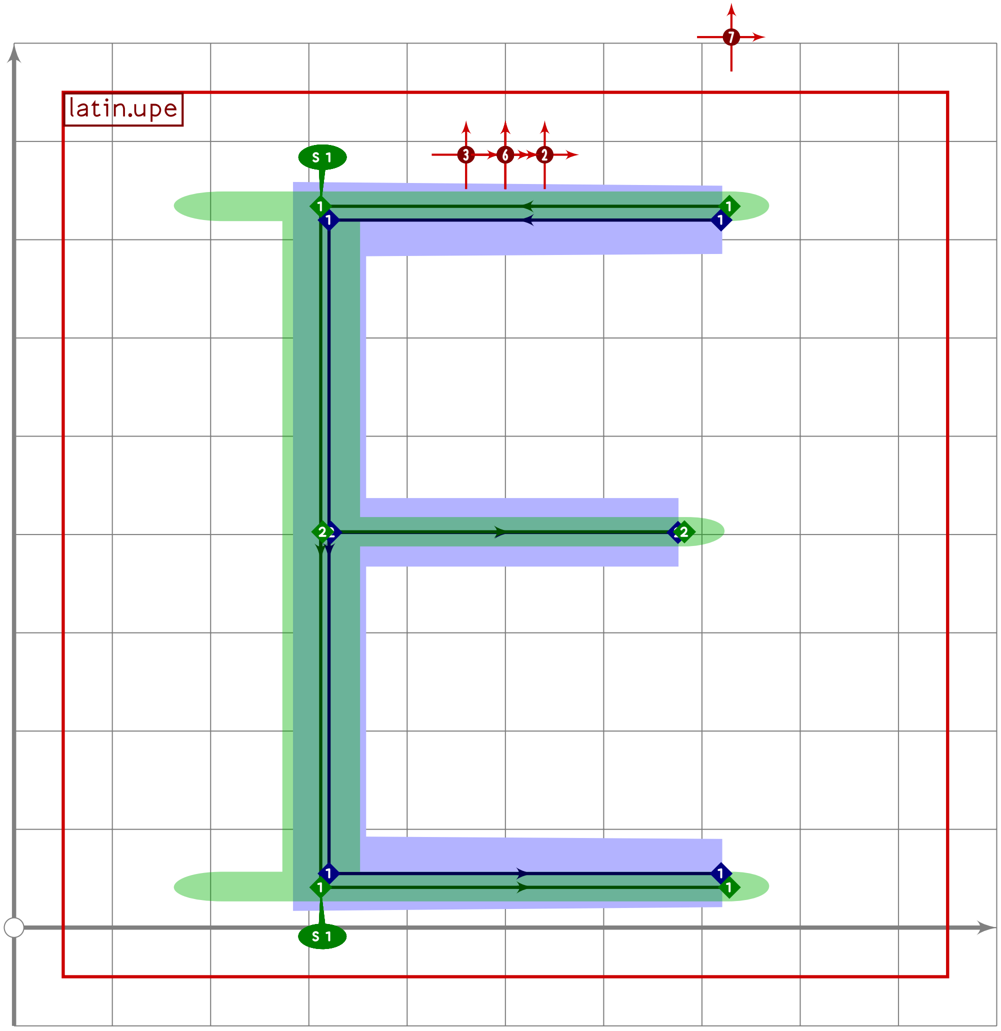


```

vardef latin.upd =
  push_pbox_toexpand(”latin.upd”);
  y1=y5=latin_wide_high_h;
  y2=y3=latin_wide_low_h;
  y4=0.52[y2,y1];

  (x1+x4)/2=510;
  (x4-x1)=0.85*(y1-y2);
  x1=x2;
  x3=x5=0.35[x1,x4];

  push_stroke(z4..{left}z5−z1−z2−z3{right}..cycle,
    (1.6,1.6)−(1.6,1.6)−(1.6,1.6)−(1.6,1.6)−(1.6,1.6)−cycle);
  set_botip(0,2,1);
  set_botip(0,3,1);
  set_boserif(0,2,1);
  set_boserif(0,3,1);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  tsu_accent.shift_anchors(true)((-50,0));
  expand_pbox;
enddef;
```

U+00D0
tsuku.Eth



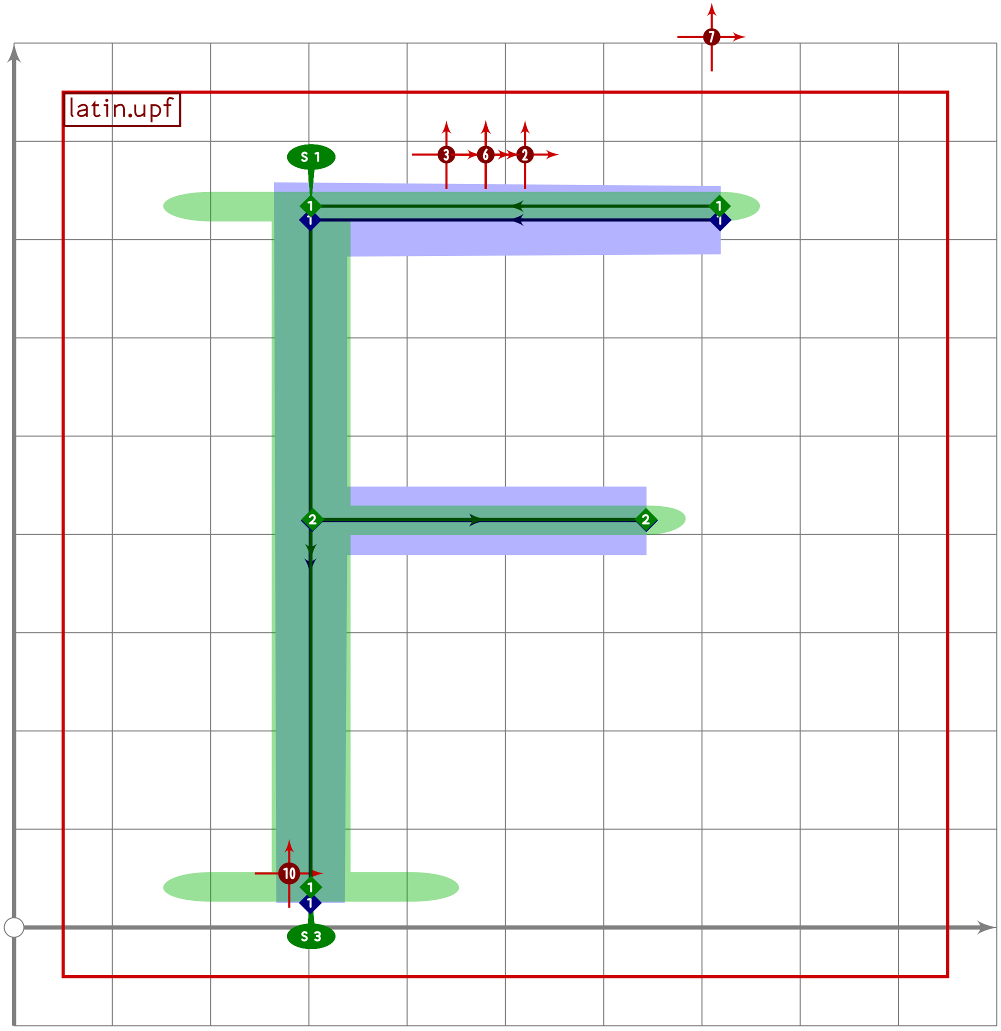
```

vardef latin.upeth =
  push_pbox_toexpand("latin.upeth");
  latin.upd;
  push_stroke((0.5[z1,z2]+(-170,0))–(0.5[z1,z2]+(220,0)),
    (1.6,1.6)–(1.6,1.6));

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  expand_pbox;
enddef;
```



```

vardef latin.upe =
  push_pbox_toexpand(”latin.upe”);
  y1=y2=latin_wide_high_h;
  y3=y4=latin_wide_low_h;
  y5=y6=vmetric(0.522);

  (x1+x2)/2=520;
  x2=x3;
  x1=x4;
  x5=x2+2;
  x6=0.89[x2,x1];
  (x1-x2)=(y2-y3)*0.6;

```

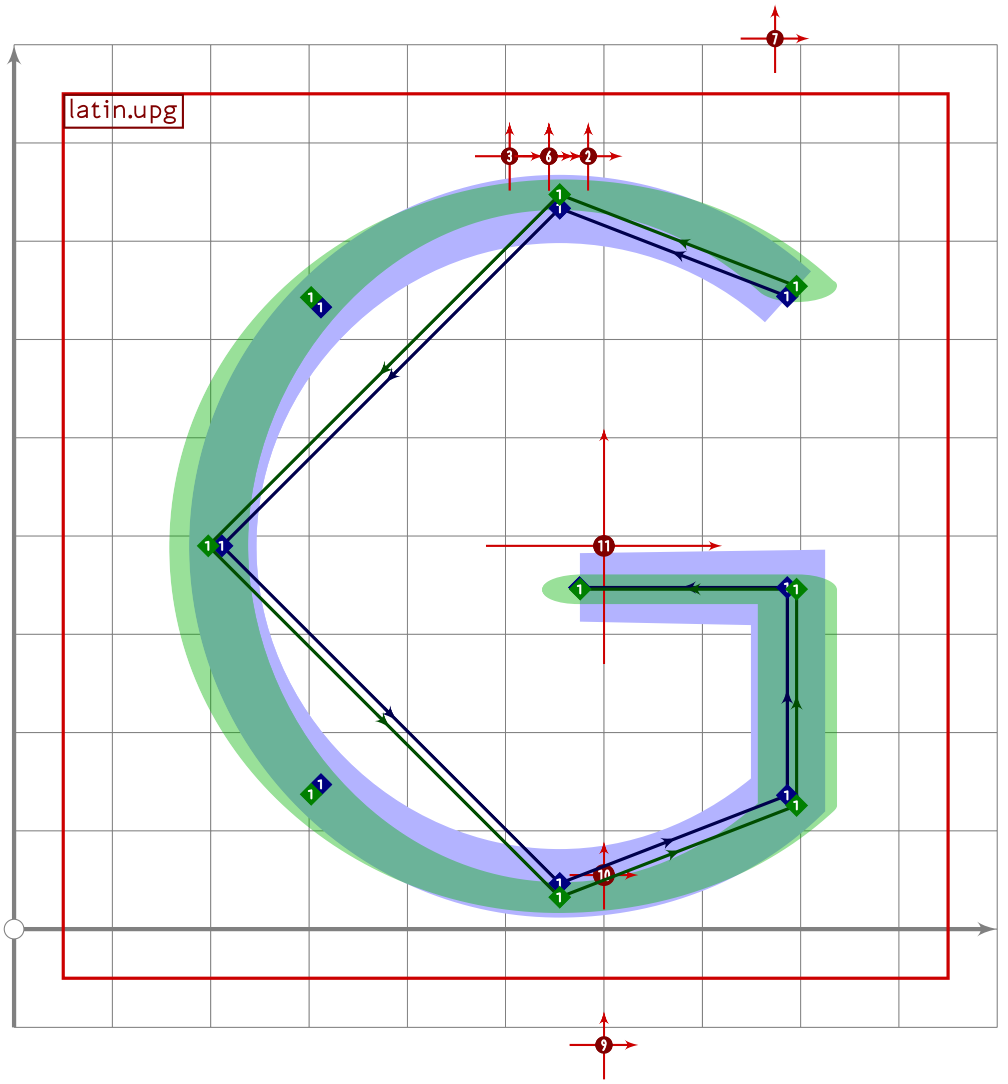
U+FF26
tsuku.uniFF26

```
  push_stroke(z1--z2--z3--z4,(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,1,1);
  set_botip(0,2,1);
  set_boserif(0,1,1);
  set_boserif(0,2,1);

  push_stroke(z5--z6,(1.6,1.6)--(1.6,1.6));

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  expand_pbox;
enddef;
```



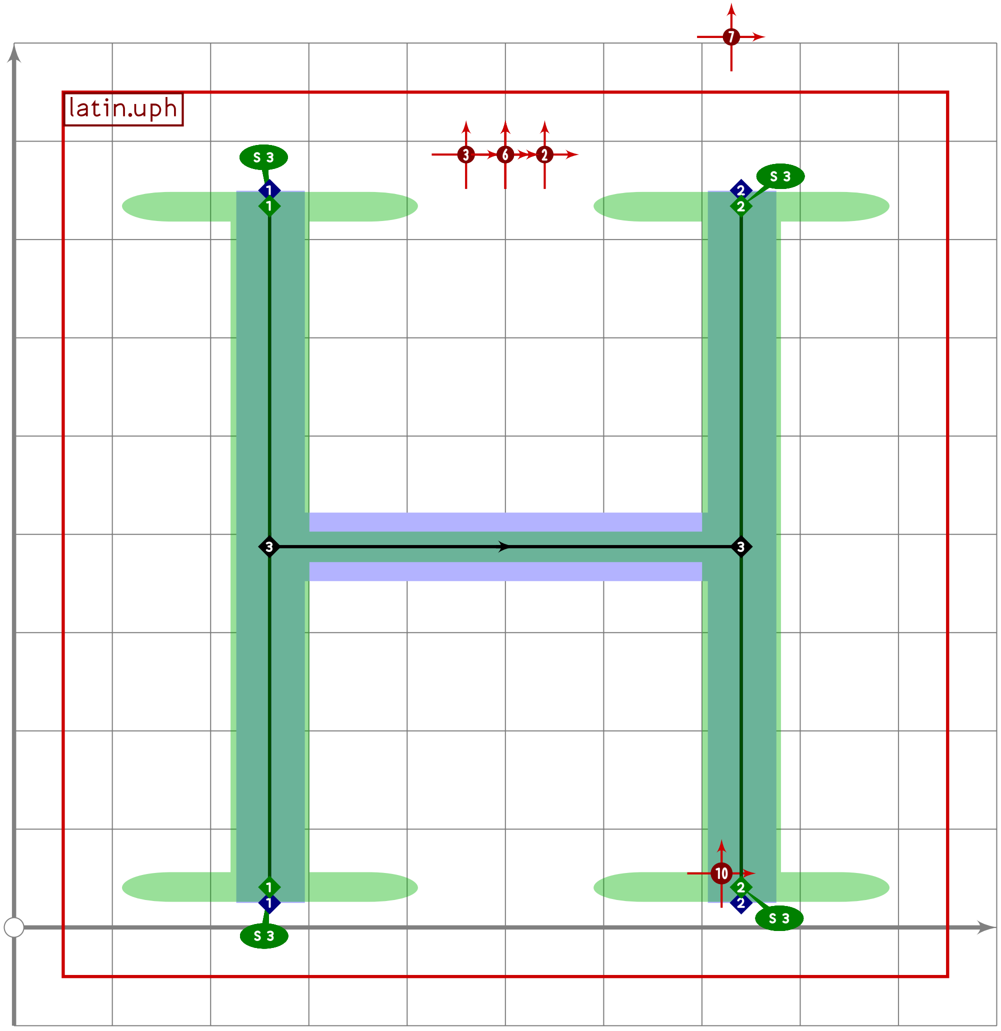
```

vardef latin.upf =
  push_pbox_toexpand("latin.upf");
  y1=y2=latin_wide_high_h;
  y3=latin_wide_low_v;
  y4=y5=vmetric(0.54);

  (x1+x2)/2=510;
  x3=x2;
  x4=x2+2;
  x5=0.82[x2,x1];
  (x1-x2)=(y2-y3)*0.6;

  push_stroke(z1--z2--z3,(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,1,1);
  set_boserif(0,1,1);
  set_boserif(0,2,3);

  push_stroke(z4--z5,(1.6,1.6)--(1.6,1.6));

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper])+(-20,0));
  tsu_accent.shift_anchors(ai=anc_lower_connect)((-220,0));
  expand_pbox;
enddef;
```

U+FF27
tsuku.uniFF27



```
vardef latin.upg =
  push_pbox_toexpand("latin.upg");
  push_stroke(
    (subpath (0.53,3.47) of ((1,0)..(0,1)..(-1,0)..(0,-1)..cycle))
       scaled ((latin_wide_high_r-latin_wide_low_r)/2)
       shifted (centre_pt+(55,0)),
    (1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)–
      (1.6,1.6)–(1.6,1.6));

  x1=xpart point 4 of get_strokep(0);
  y1=y2=vmetric(0.44);
  x1-x2=y1-(ypart point 4 of get_strokep(0))*1.0;

  replace_strokep(0)(oldp–z1–z2);
  set_botip(0,4,1);
  set_botip(0,5,1);


  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
```

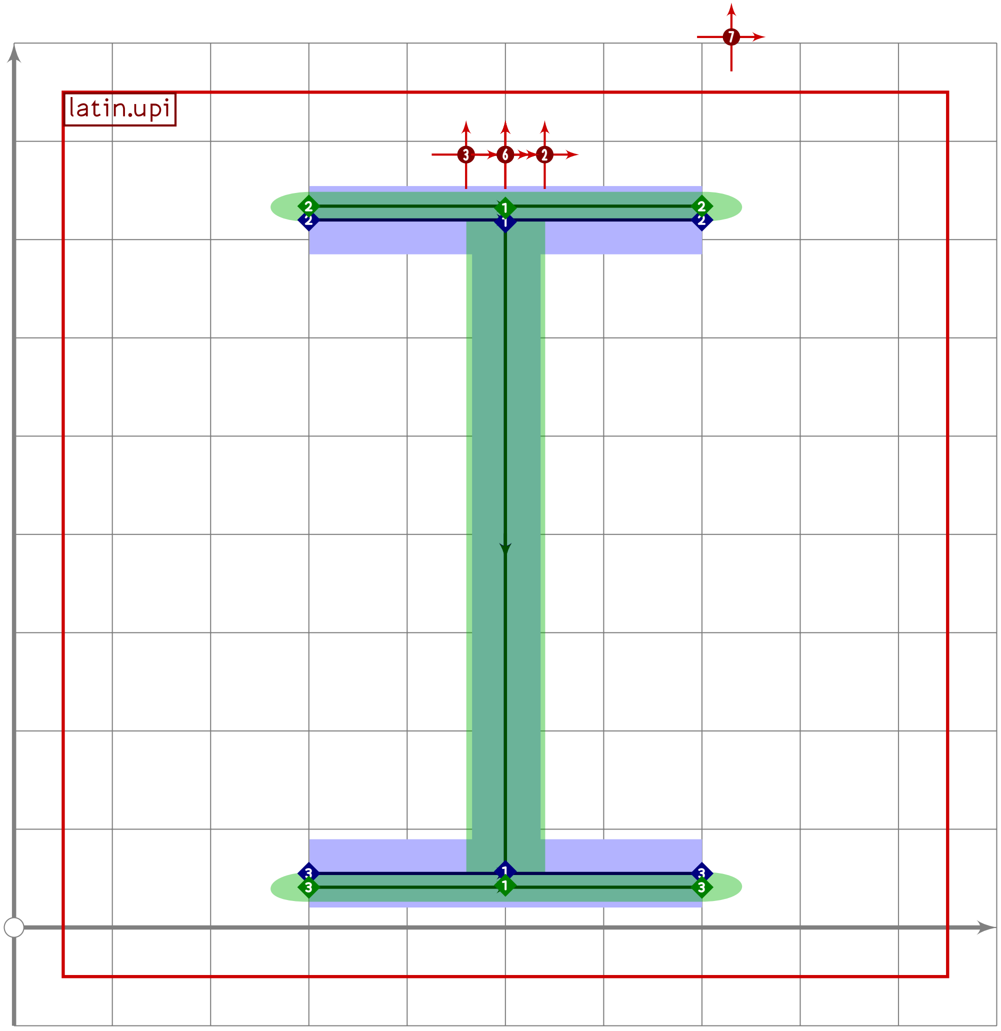
```
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper])+(44,0));
  tsu_accent.shift_anchors(ypart olda<vmetric(0.52))((100,0));
  expand_pbox;
enddef;
```



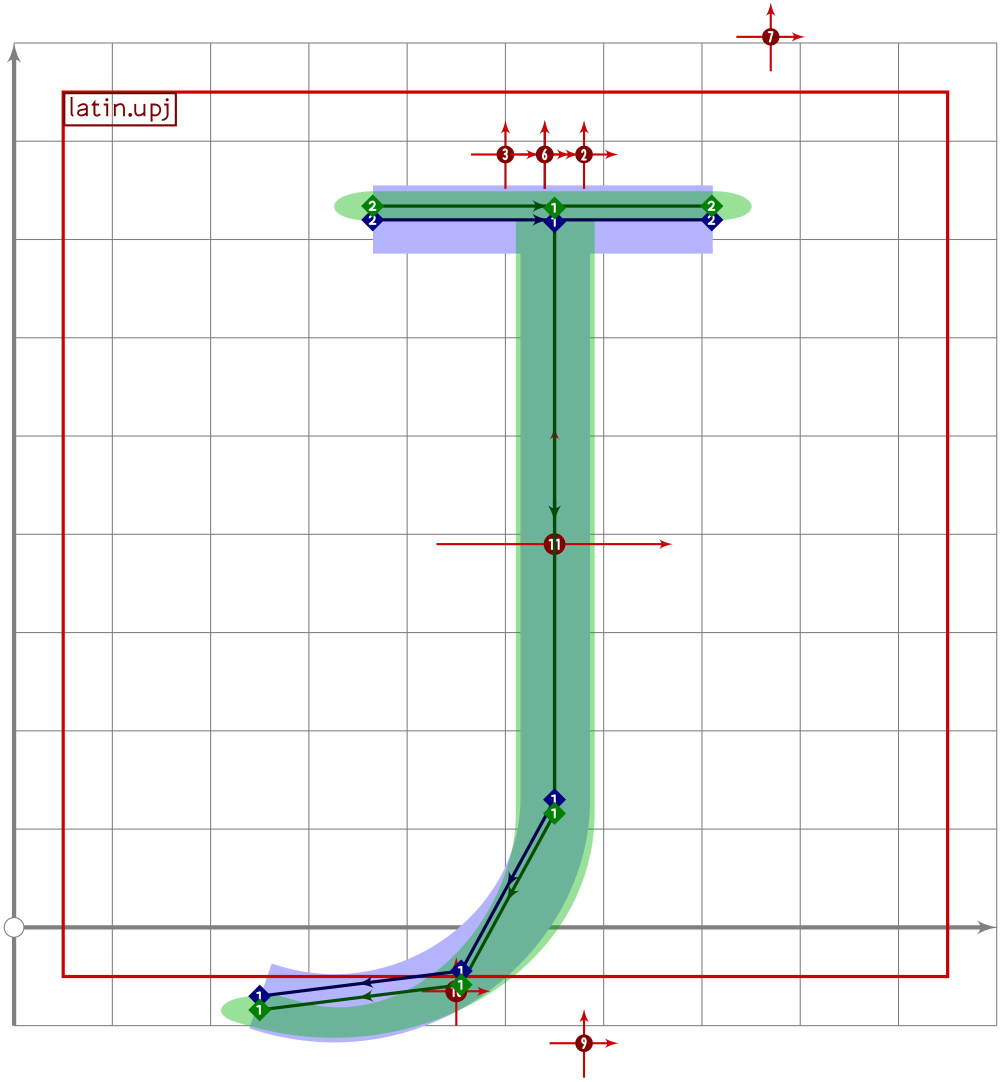
```

vardef latin.uph =
  push_pbox_toexpand(”latin.uph”);
  z1=(260,latin_wide_high_v);
  z2=(740,latin_wide_high_v);
  z3=(260,latin_wide_low_v);
  z4=(740,latin_wide_low_v);

  z5=whatever[z1,z3];
  z6=whatever[z2,z4];
  y5=y6=vmetric(0.5);

  push_stroke(z1—z3,(1.6,1.6)—(1.6,1.6));
  set_boserif(0,0,3);
  set_boserif(0,1,3);
```

U+FF29
tsuku.uniFF29

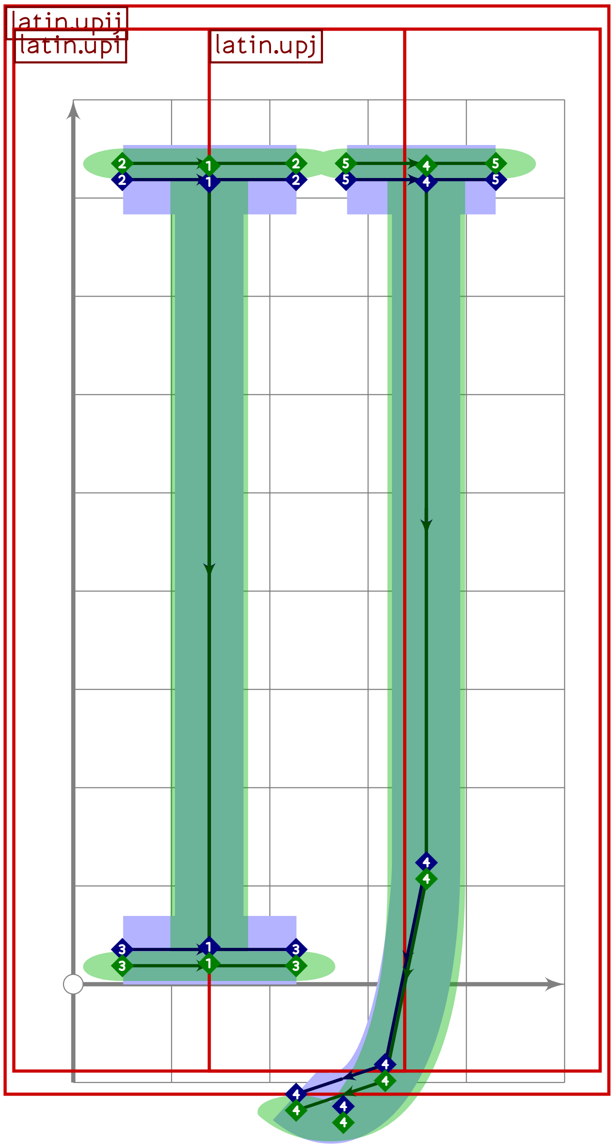
```

  push_stroke(z2--z4,(1.6,1.6)--(1.6,1.6));
  set_boserif(0,0,3);
  set_boserif(0,1,3);

  push_stroke(z5--z6,(1.6,1.6)--(1.6,1.6));

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  tsu_accent.shift_anchors(ai=anc_lower_connect)((220,0));
  expand_pbox;
enddef;
```



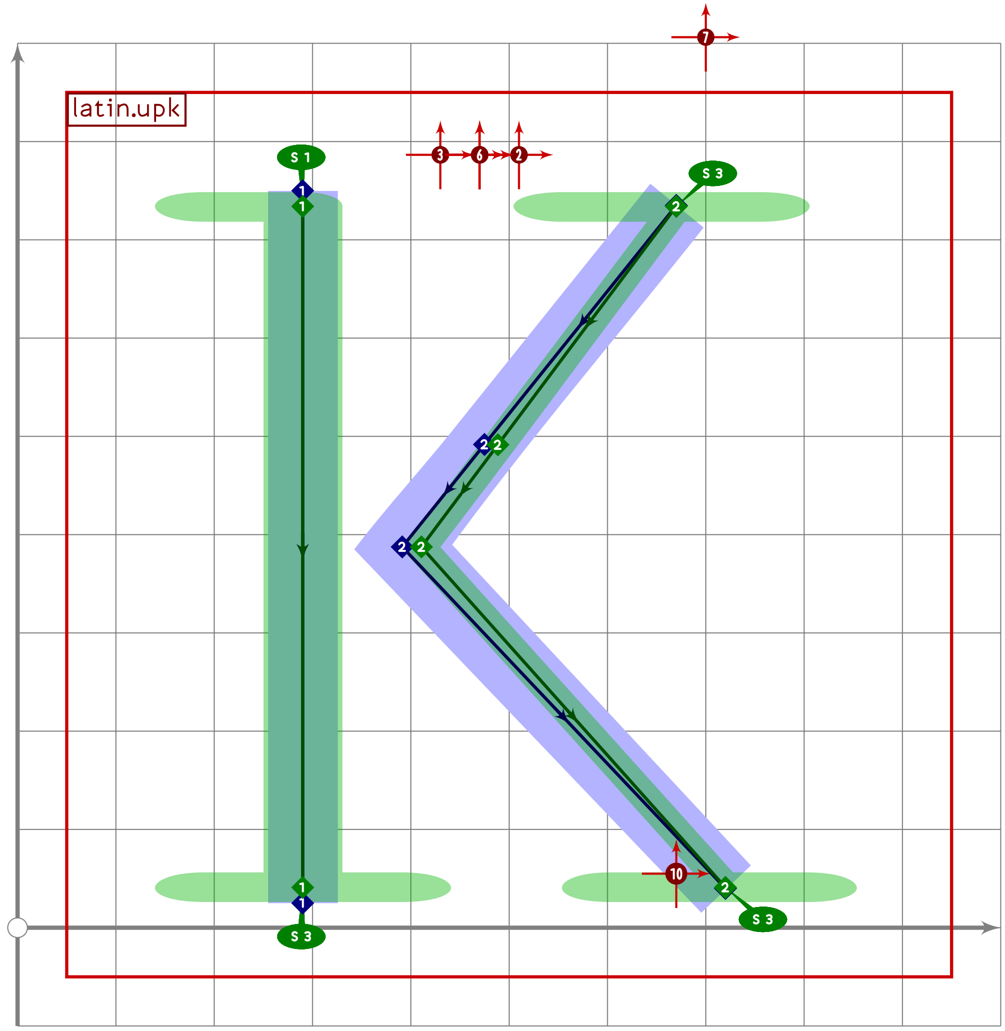
```

vardef latin.upi =
  push_pbox_toexpand("latin.upi");
  push_stroke((500,latin_wide_high_h-2)--(500,latin_wide_low_h+2),
    (1.6,1.6)--(1.6,1.6));

  push_stroke((300,latin_wide_high_h)--(700,latin_wide_high_h),
    (1.6,1.6)--(1.6,1.6));

  push_stroke((300,latin_wide_low_h)--(700,latin_wide_low_h),
    (1.6,1.6)--(1.6,1.6));

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  expand_pbox;
enddef;
```



```

vardef latin.upj =
  push_pbox_toexpand(”latin.upj”);
  z1=(550,latin_wide_high_h-2);
  z2=(550,latin_wide_low_h+75);
  z3=z2+(-300,-200);

  push_stroke((z1--z2)..{curl 0.8}z3,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(insert_nodes(oldp)(1.5));

  push_stroke((365,latin_wide_high_h)--(710,latin_wide_high_h),
    (1.6,1.6)--(1.6,1.6));

  tsu_accent.shift_anchors(true)((50,0));

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper])+(-10,0));
  tsu_accent.shift_anchors(ai=anc_lower)((30,0));
  tsu_accent.shift_anchors(ai=anc_lower_connect)((-100,-120));
  expand_pbox;
enddef;
```



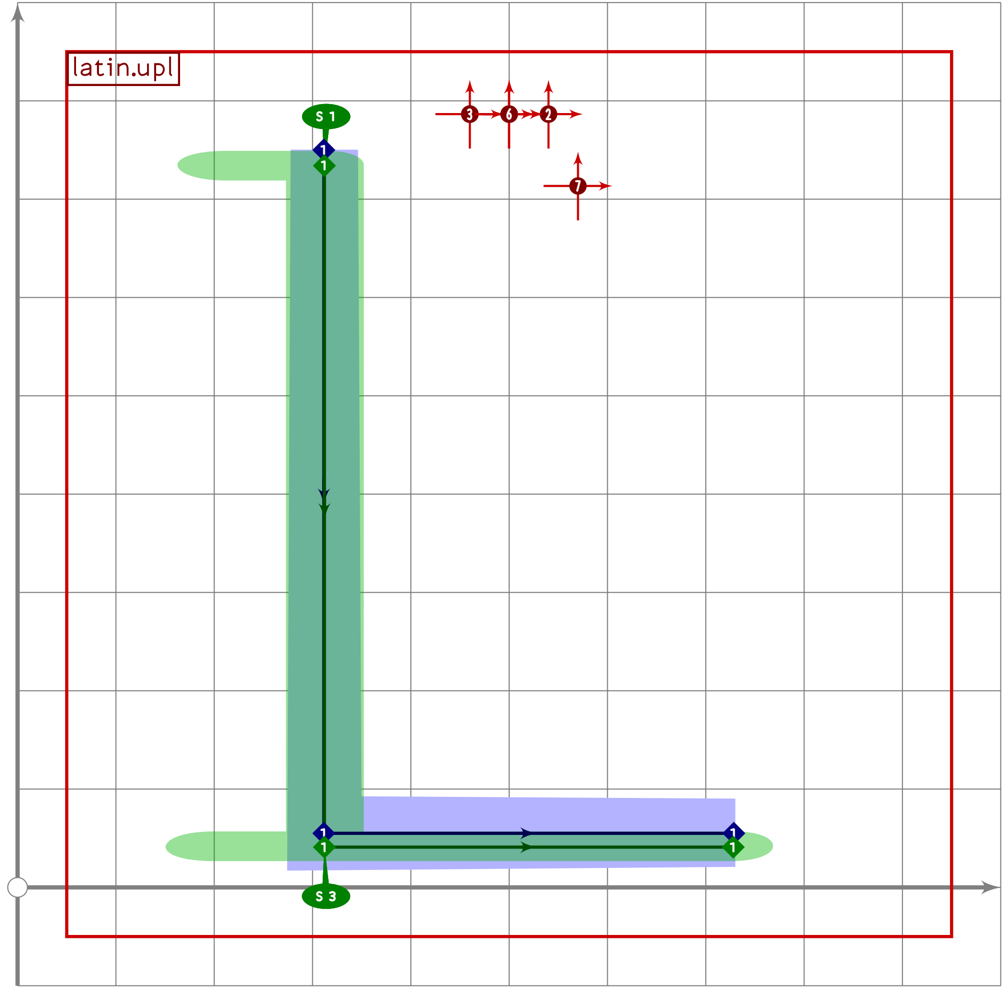
```

vardef latin.upij =
  push_pbox_toexpand("latin.upij");
  tsu_xform(identity shifted (-250,0))(latin.upi);
  tsu_xform(identity shifted (200,0))(latin.upj);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  expand_pbox;
enddef;
```



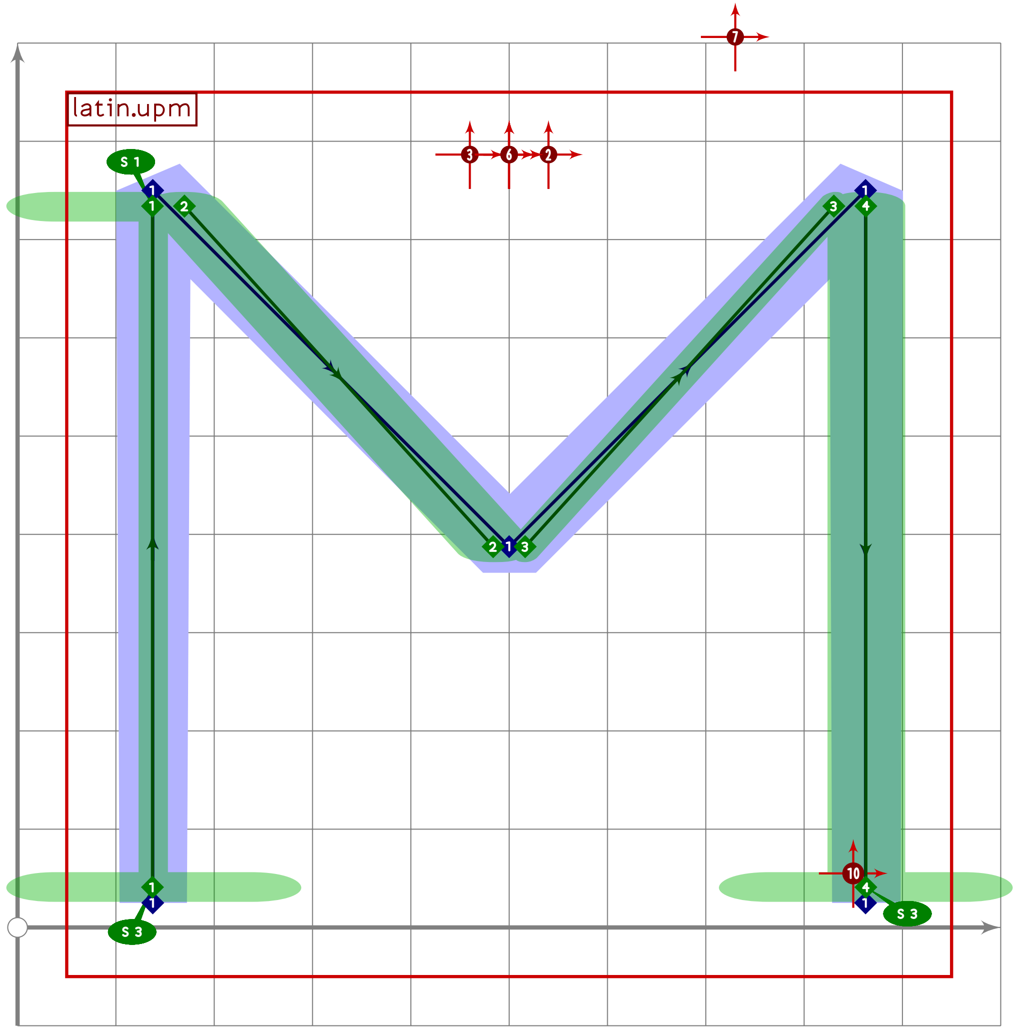
```

vardef latin.upk =
  push_pbox_toexpand("latin.upk");
  z1=(290,latin_wide_high_v);
  z2=(290,latin_wide_low_v);
  z3=(670,0.5[latin_wide_high_h,latin_wide_high_v]);
  x4=290+mbrush_width*if sharp_corners: 2.7 else: 2.3 fi;
  y4=vmetric(0.5);
  z5=(720,0.5[latin_wide_low_h,latin_wide_low_v]);

  push_stroke(z1--z2,(1.6,1.6)--(1.6,1.6));
```

U+FF2C
tsuku.uniFF2C

```
    set_boserif(0,0,1);
    set_boserif(0,1,3);

    push_stroke(z3--(0.7[z3,z4])--z4--z5,
      (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
    set_botip(0,2,1);
    if do_alternation:
      set_boalternate(0);
      set_boserif(0,0,3);
      set_boserif(0,3,3);
    fi;

    tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
      (((0,0) transformed tsu_xf.cap_upper_accent)-
       ((0,0) transformed accent_default[anc_upper])+(-30,0));
    tsu_accent.shift_anchors(ai=anc_lower_connect)((170,0));
    expand_pbox;
  enddef;
```

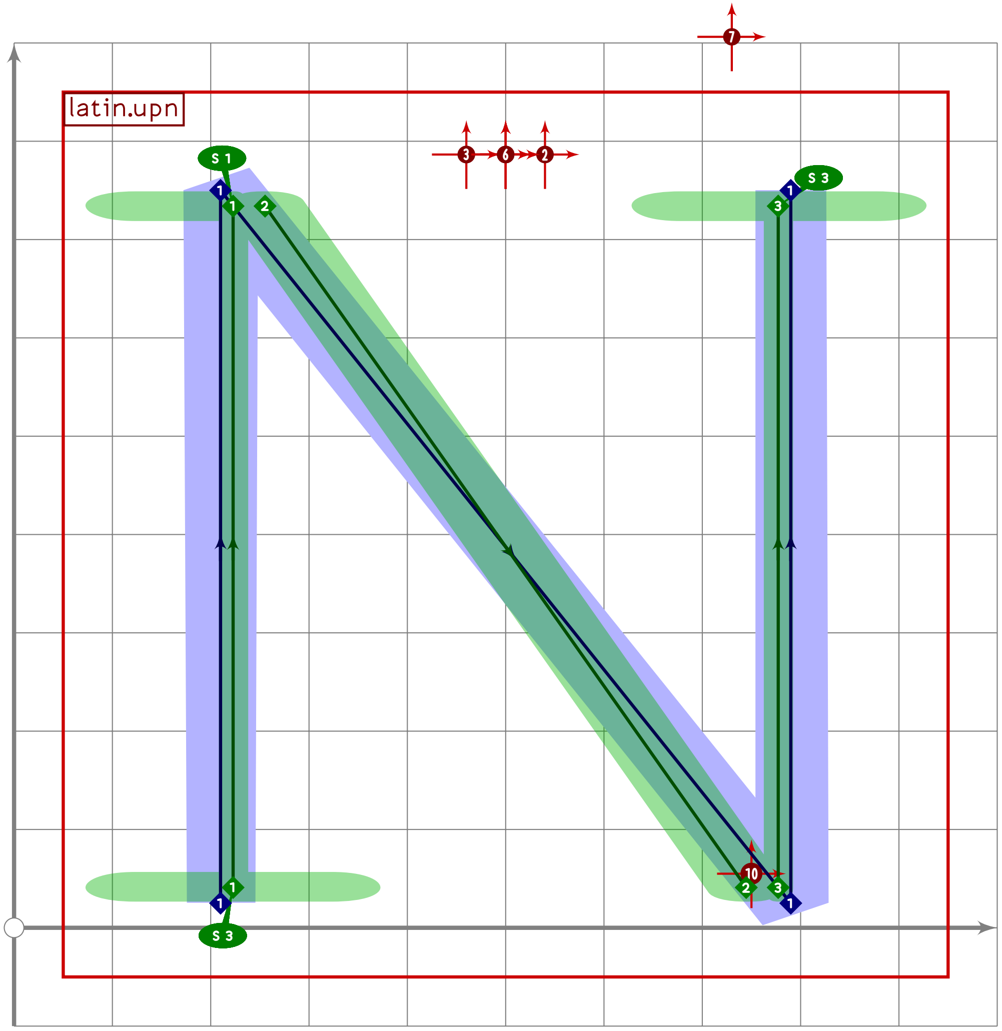


```

  vardef latin.upl =
    push_pbox_toexpand(”latin.upl”);
  y1=latin_wide_high_v;
  y2=y3=latin_wide_low_h;
  x1=x2;
  (x1+x3)/2=520;
  (x3-x1)=(y1-y2)*0.6;

  push_stroke(z1--z2--z3,(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,1,1);
  set_boserif(0,0,1);
  set_boserif(0,1,3);

  tsu_accent.shift_anchors((ypart olda>vmetric(0.52))
                           and not (ai=anc_caron_comma))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  tsu_accent.shift_anchors(ai=anc_caron_comma)((-160,40));
  expand_pbox;
enddef;
```



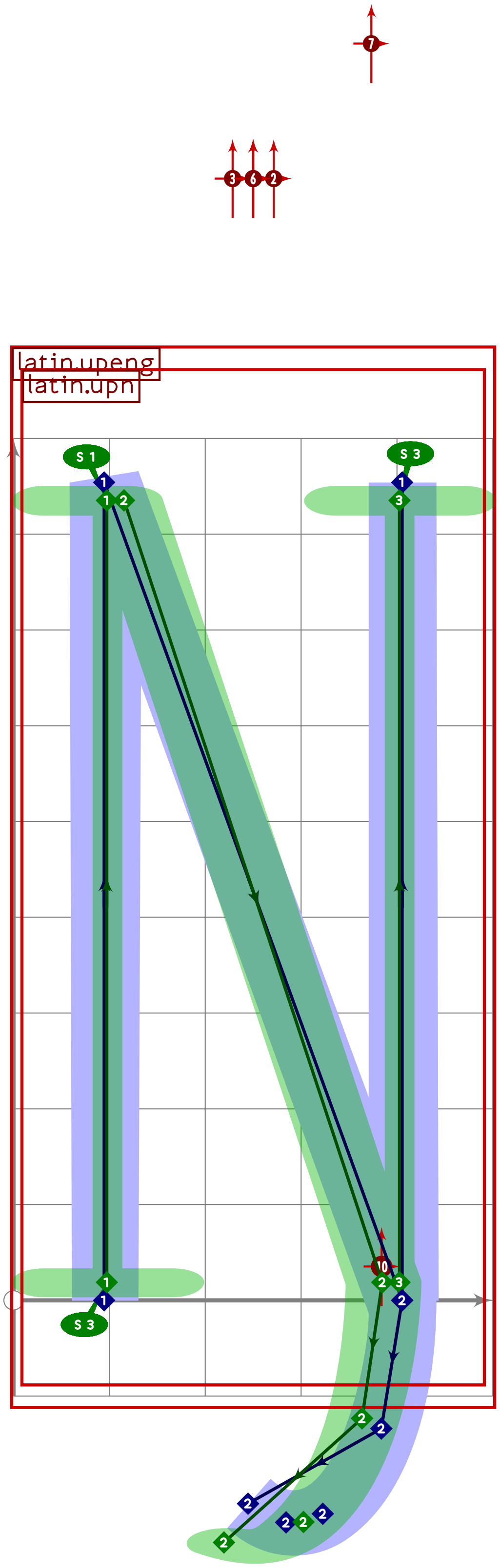
```

vardef latin.upm =
push_pbox_toexpand(”latin.upm”);
y1=y5=latin_wide_low_v;
y2=y4=latin_wide_high_v;
y3=(y1+y2)/2;

if do_alternation:
  x1=x2;
  x3=500+alternate_adjust/2;
  x4=x5;
  (x3-x1)=(x5-x3);

  (x5-x1)=(y2-y1);

  push_stroke((z1--z2) shifted (alternate_adjust*left),
    (1.6,1.6)--(1.6,1.6));
  set_boserif(0,0,3);
  set_boserif(0,1,1);
  set_boalternate(0);

  push_stroke(z2--(z3+alternate_adjust*left),(1.6,1.6)--(1.6,1.6));

  push_stroke(z3--(z4+alternate_adjust*left),(1.6,1.6)--(1.6,1.6));
  set_boalternate(0);

  push_stroke(z4--z5,(1.6,1.6)--(1.6,1.6));
  set_boserif(0,1,3);
else:
  x1=x2;
  x3=500;
  x4=x5;
  (x3-x1)=(x5-x3);

  (x5-x1)=(y2-y1);

  push_stroke(z1--z2--z3--z4--z5,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,1,0);
  set_botip(0,2,0);
  set_botip(0,3,0);
  set_boserif(0,0,3);
  set_boserif(0,1,1);
  set_boserif(0,4,3);
fi;

tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
  (((0,0) transformed tsu_xf.cap_upper_accent)-
   ((0,0) transformed accent_default[anc_upper]));
tsu_accent.shift_anchors(ai=anc_lower_connect)((350,0));
```

```
  expand_pbox;
enddef;
```



```

vardef latin.upn =
  push_pbox_toexpand(”latin.upn”);
  y1=y3=latin_wide_low_v;
  y2=y4=latin_wide_high_v;

  x1=x2;
  x3=x4;
  (x1+x3)/2=500;
  (x3-x1)=(y2-y1)*4/5;

  if do_alternation:
    push_stroke(z1–z2,(1.6,1.6)–(1.6,1.6));
    set_boserif(0,0,3);
    set_boserif(0,1,1);
    set_boalternate(0);

    push_stroke((z2+alternate_adjust*right)–(z3+alternate_adjust*left),
      (1.6,1.6)--(1.6,1.6));

    push_stroke(z3--z4,(1.6,1.6)--(1.6,1.6));
    set_boserif(0,1,3);
    set_boalternate(0);
  else:
    push_stroke(z1--z2--z3--z4,(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
    set_botip(0,1,0);
    set_botip(0,2,0);
    set_boserif(0,0,3);
    set_boserif(0,1,1);
    set_boserif(0,3,3);
  fi;

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  tsu_accent.shift_anchors(ai=anc_lower_connect)((250,0));
  expand_pbox;
enddef;
```

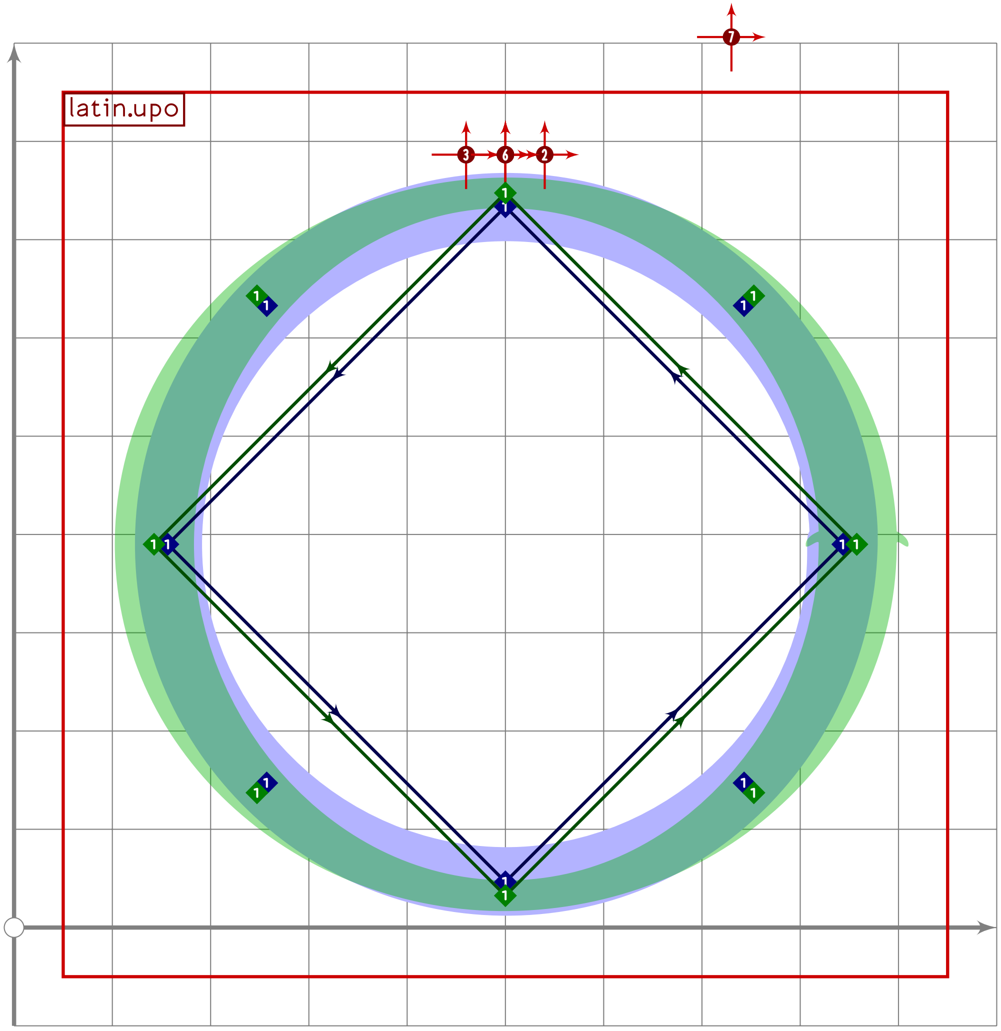
U+014A
tsuku.Eng



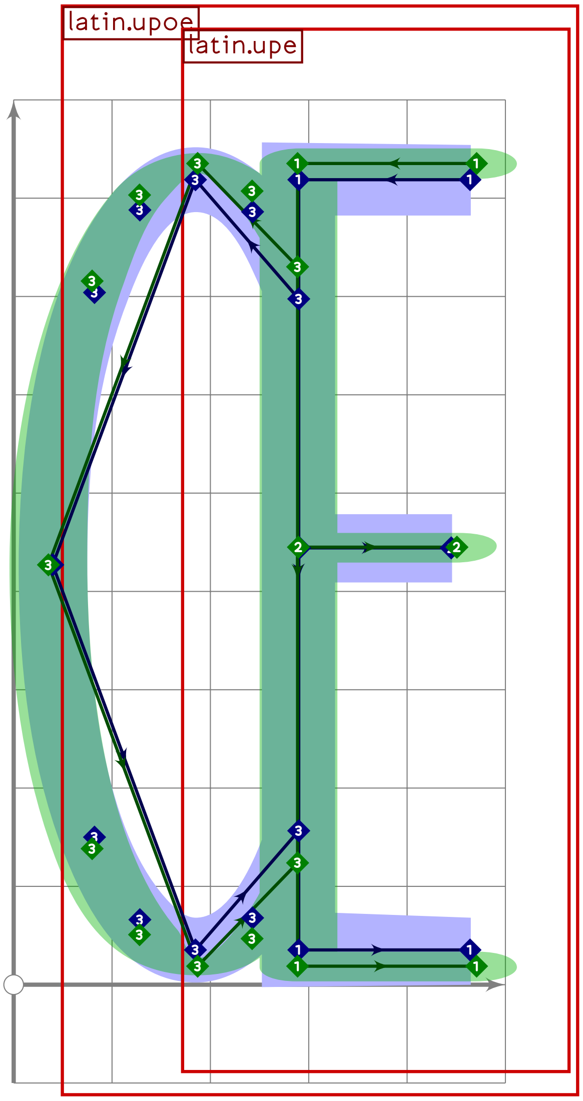
```
vardef latin.upeng =
  push_pbox_toexpand(”latin.upeng”);
  latin.upn;
  y5=latin_wide_desc_h;
```

U+FF2F
tsuku.uniFF2F

```
  if do_alternation:
    x5=x3-alternate_adjust-300;
    replace_strokep(-1)(oldp{dir 268}..{curl 0.8}z5);
    replace_strokep(-1)(insert_nodes(oldp)(1.3));
    replace_strokeq(-1)(oldq--(1.6,1.6)--(1.6,1.6));
  else:
    x5=x3-300;
    push_stroke(z3{dir 268}..{curl 0.8}z5,(1.6,1.6)--(1.6,1.6));
    replace_strokep(0)(insert_nodes(oldp)(0.3));
  fi;

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  expand_pbox;
enddef;
```



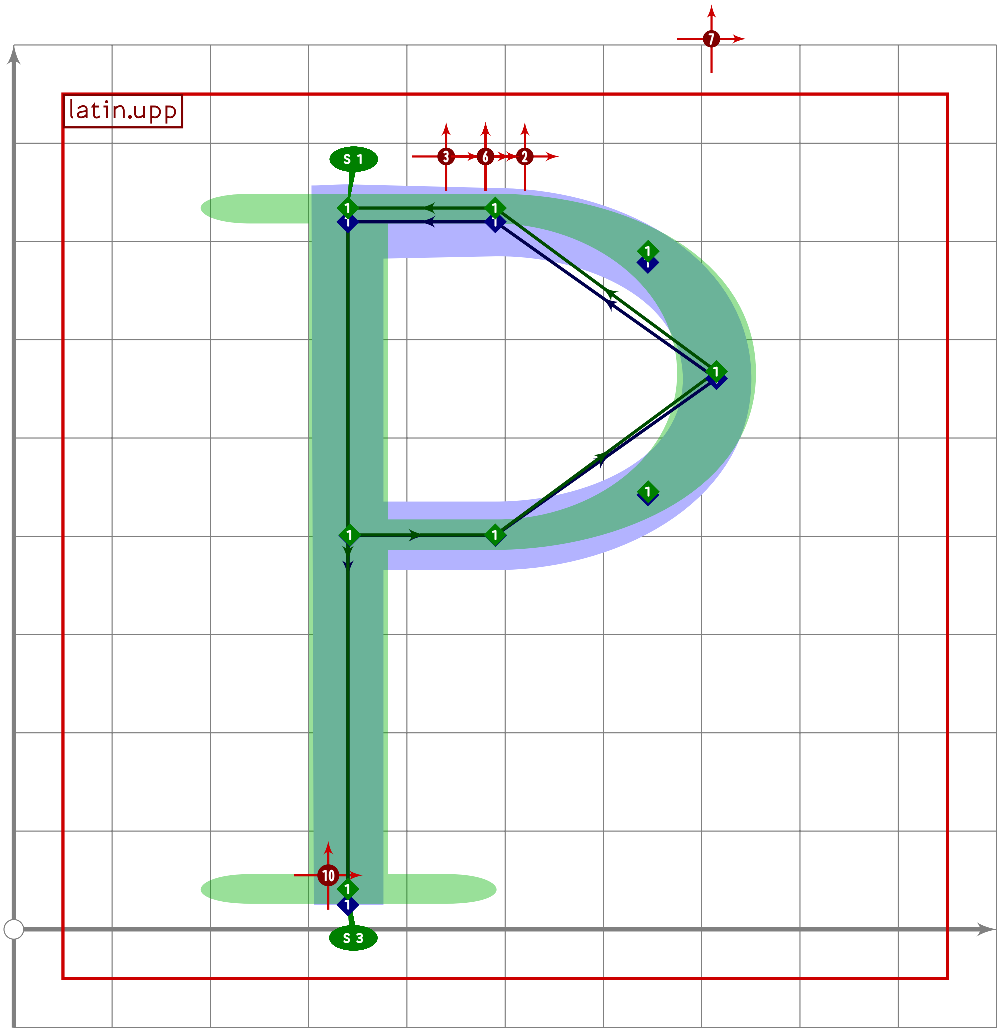
```

vardef latin.upo =
  push_pbox_toexpand("latin.upo");
  push_stroke(
    ((1,0)..(0,1)..(-1,0)..(0,-1)..cycle)
    scaled ((latin_wide_high_r-latin_wide_low_r)/2)
    shifted centre_pt,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--cycle);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  expand_pbox;
enddef;
```



```

vardef latin.upoe =
  push_pbox_toexpand("latin.upoe");
  tsu_xform(identity shifted (280,0))(latin.upe);
  set_boserif(-1,1,whatever);
  set_boserif(-1,2,whatever);
  push_stroke(
    ((1,0)..(0,1)..(-1,0)..(0,-1)..(1,0))
```

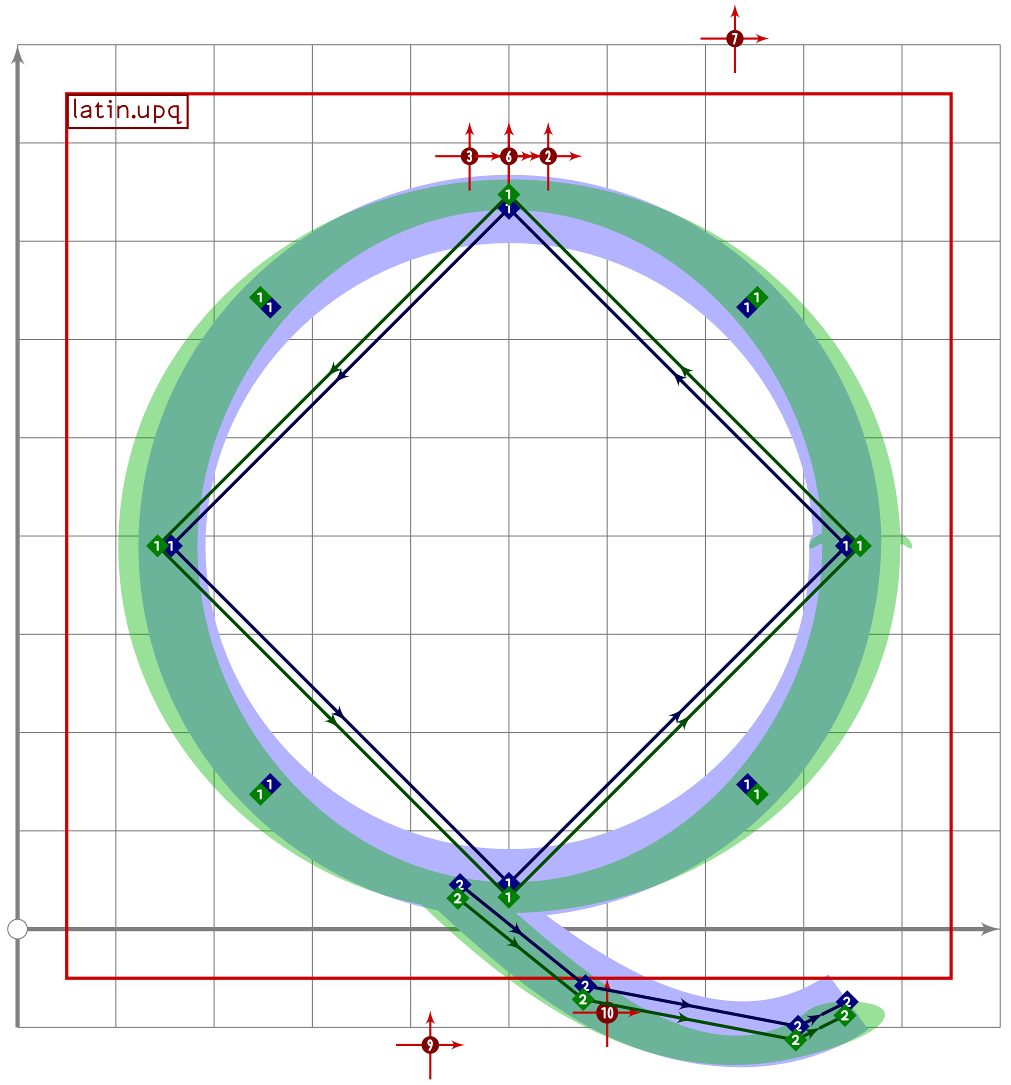
LATI

```
       scaled ((latin_wide_high_h-latin_wide_low_h)/2)
       shifted (360,0.5[latin_wide_high_h,latin_wide_low_h]),
    (1.2,1.2)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.2,1.2));
  replace_strokep(0)(
    subpath (xpart (oldp intersectiontimes get_strokep(-2)),
             4-xpart ((reverse oldp) intersectiontimes get_strokep(-2)))
    of oldp);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  expand_pbox;
enddef;

vardef latin.upp_base(expr b) =
  x1=x5+2;
  x5=340;
  x2=x4=x5+b*0.4;
  x3=x5+b;

  y1=y2=vmetric(0.52);
  y3=(y2+y4)/2;
  y4=y5=latin_wide_high_h;

  push_stroke(z1--z2{right}..z3..{left}z4--z5,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
enddef;
```

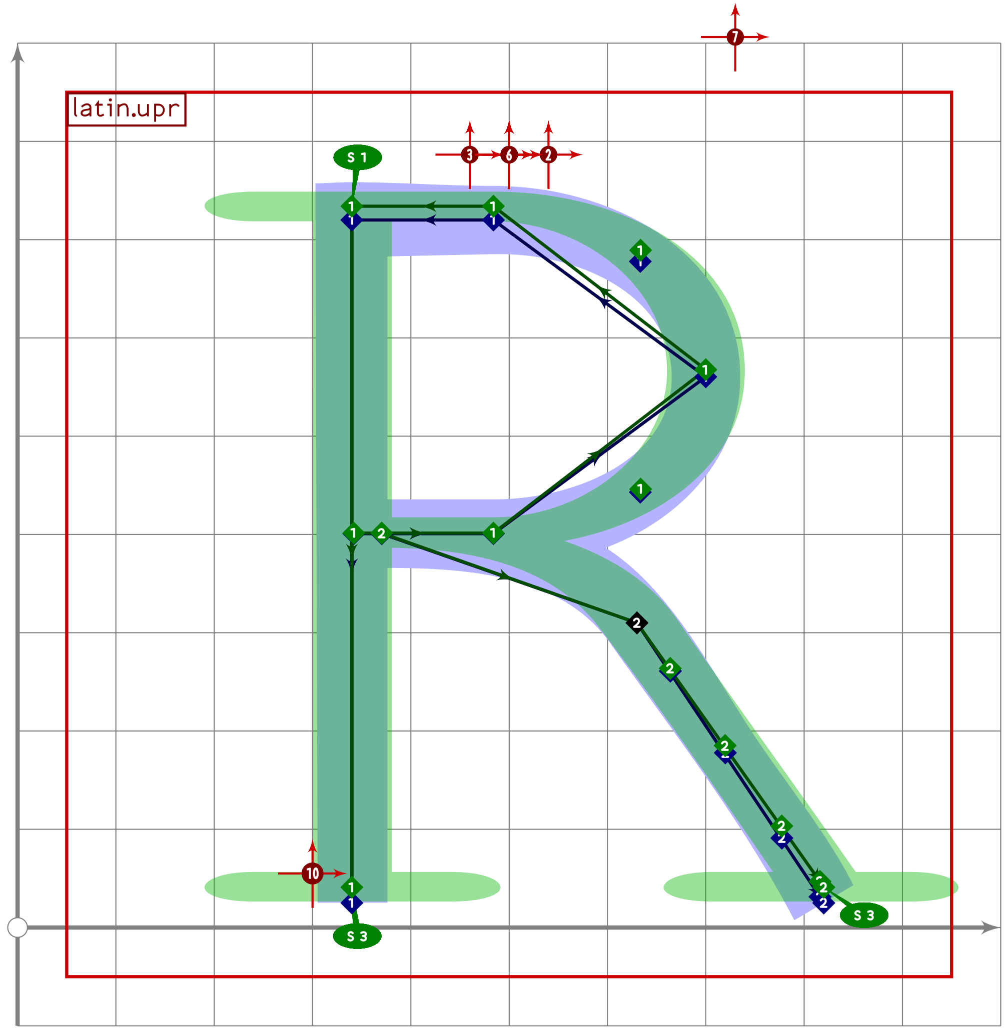


```

vardef latin.upp =
  push_pbox_toexpand(”latin.upp”);
  latin.upp_base(375);
  replace_strokep(0)(oldp-(xpart point infinity of oldp,latin_wide_low_v));
  replace_strokeq(0)(oldq-(1.6,1.6));
  set_botip(0,4,1);
  set_boserif(0,4,1);
  set_boserif(0,5,3);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper])+(-20,0));
  tsu_accent.shift_anchors(ai=anc_lower_connect)((-180,0));
  expand_pbox;
enddef;
```

U+FF31
tsuku.uniFF31



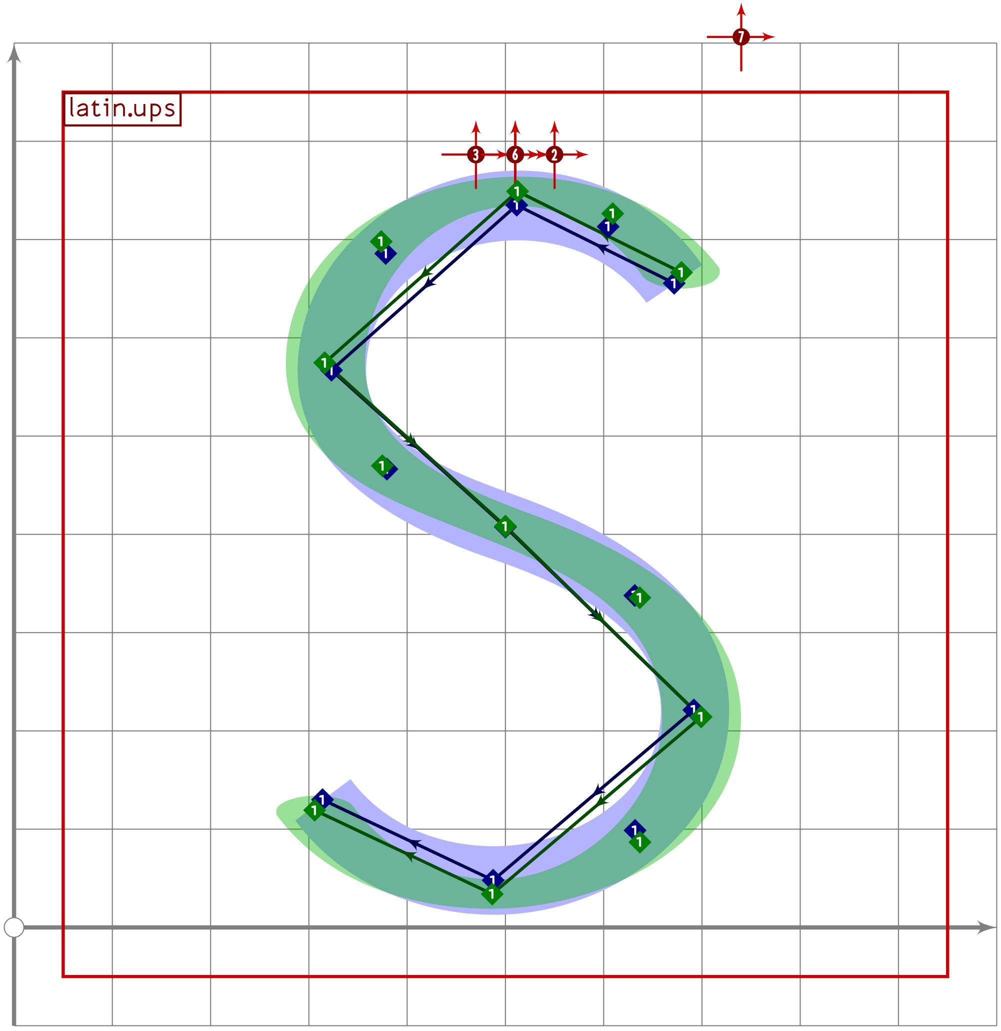
```
vardef latin.upq =
  push_pbox_toexpand(”latin.upq”);
  push_stroke(((1,0)..(0,1)..(-1,0)..(0,-1)..cycle)
    scaled ((latin_wide_high_r-latin_wide_low_r)/2)
    shifted centre_pt,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--cycle);

  z1=point 2.9 of get_strokep(0);
  z2=z1+(350,-150);
  z3=z2+(50,25);
  push_stroke(subpath (0.02,2) of (z1{curl 0}..z2..z3),
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(insert_nodes(oldp)(0.4));

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  tsu_accent.shift_anchors(ai=anc_lower)((-80,0));
  tsu_accent.shift_anchors(ai=anc_lower_connect)((100,-140));
  expand_pbox;
enddef;
```



```

vardef latin.upr =
  push_pbox_toexpand("latin.upr");
  latin.upp_base(360);
  replace_strokep(0)(oldp−(xpart point infinity of oldp,latin_wide_low_v));
  replace_strokeq(0)(oldq−(1.6,1.6));
  set_botip(0,4,1);
  set_boserif(0,4,1);
  set_boserif(0,5,3);

  push_stroke((point 0.2 of get_strokep(0)){right}..(630,310)..
    tension 3..(820,latin_wide_low_v),
    (1.6,1.6)−(1.6,1.6)−(1.6,1.6)−(1.6,1.6));
  replace_strokep(0)(insert_nodes(oldp)(1.95));
  set_boserif(0,3,3);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
```

U+FF33
tsuku.uniFF33

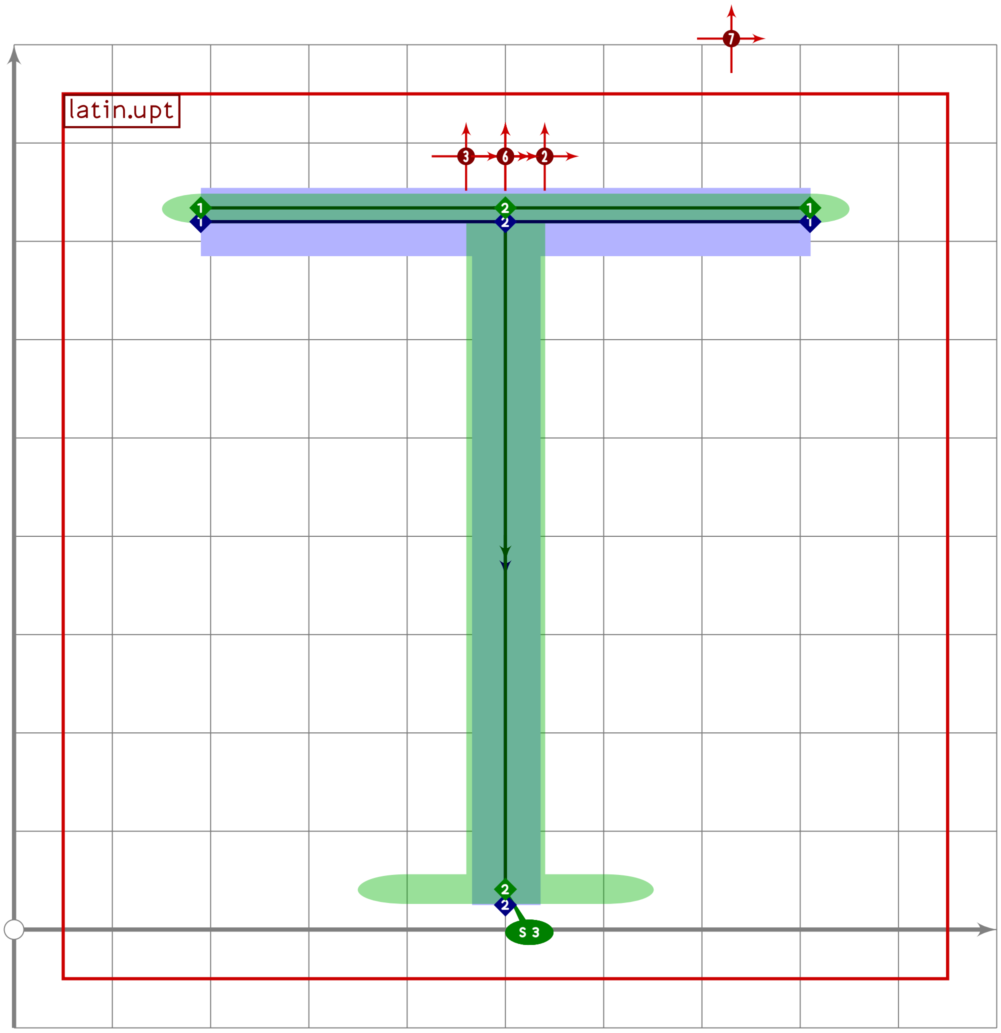
```
     (((0,0) transformed tsu_xf.cap_upper_accent)-
      ((0,0) transformed accent_default[anc_upper]));
   tsu_accent.shift_anchors(ai=anc_lower_connect)((-200,0));
   expand_pbox;
enddef;
```



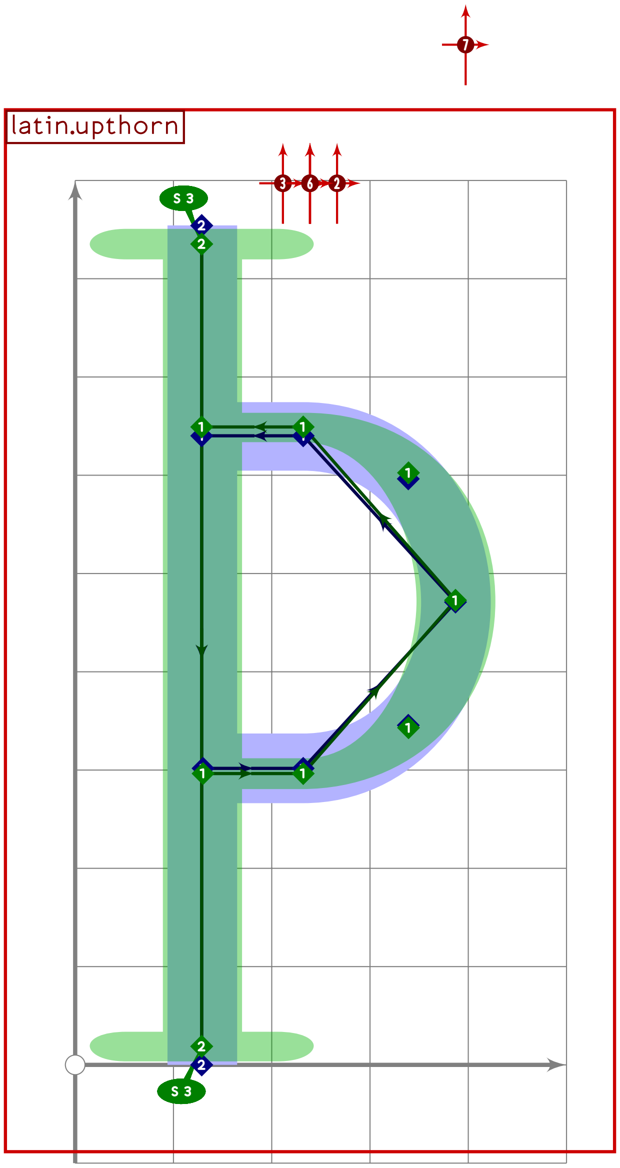
```

vardef latin.ups =
   push_pbox_toexpand("latin.ups");
   transform ta,tb;
   path mycurve;

   mycurve:=(1,0)..(0,1)..(-1,0);

   y2=latin_wide_high_r;
   y0=y3=vmetric(0.77);
   y4=vmetric(0.53);
   y5=y8=vmetric(0.25);
   y6=latin_wide_low_r;

   0.48[x1,x7]=0.48[x2,x6]=0.48[x3,x5]=x4=500;
```

```
  x5-x1=20;
  x5-x7=(y2-y6)*0.55;

  (point 0 of mycurve) transformed ta=z0;
  (point 0.35 of mycurve) transformed ta=z1;
  (point 1 of mycurve) transformed ta=z2;
  (point 2 of mycurve) transformed ta=z3;
  xypart ta=0;

  (point 0 of mycurve) transformed tb=z8;
  (point 0.35 of mycurve) transformed tb=z7;
  (point 1 of mycurve) transformed tb=z6;
  (point 2 of mycurve) transformed tb=z5;

  mycurve:=subpath (0.35,2) of mycurve;

  push_stroke((mycurve transformed ta)..z4..(reverse mycurve transformed tb),
    (1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)
      –(1.6,1.6));

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper])+(10,0));
  expand_pbox;
enddef;
```

U+FF34
tsuku.uniFF34



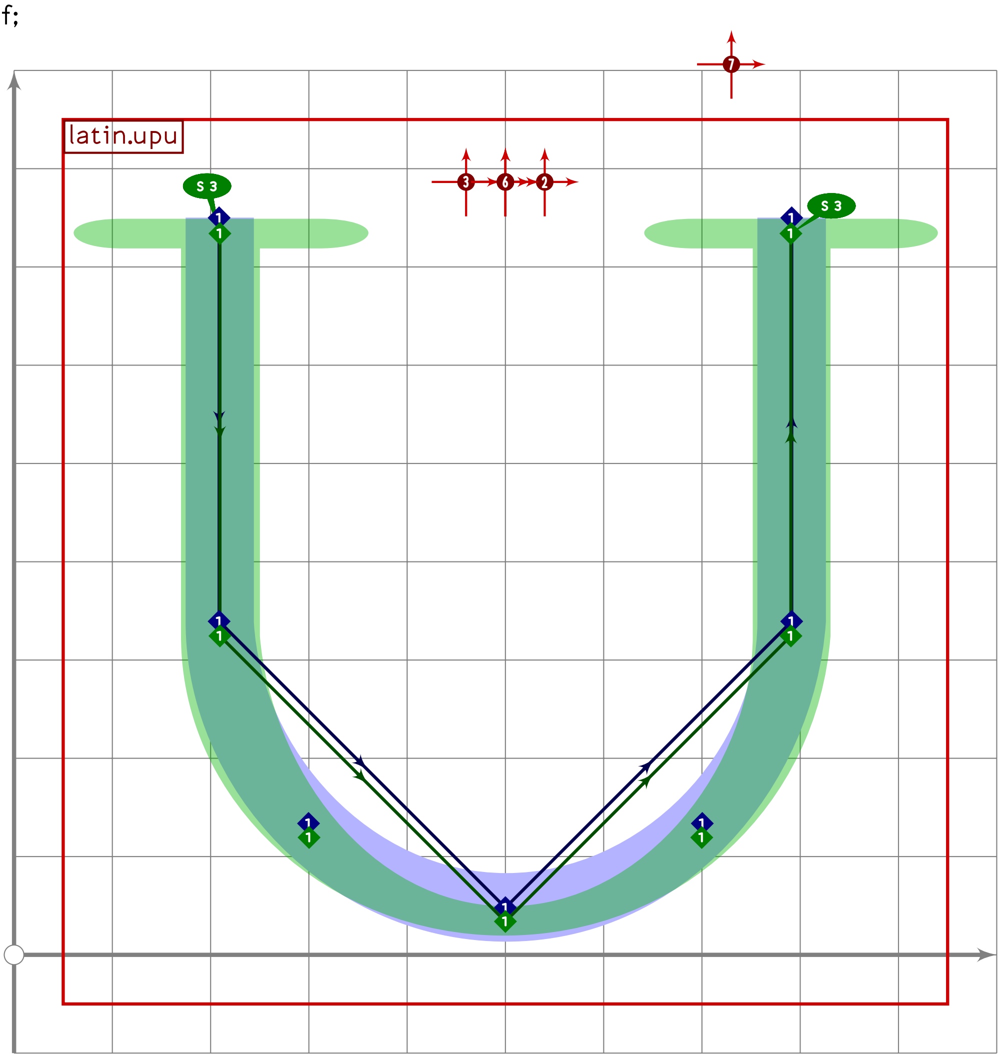
```
vardef latin.upt =
  push_pbox_toexpand(”latin.upt”);
  z1=(190,latin_wide_high_h);
  z2=(500,latin_wide_high_h);
  z3=(810,latin_wide_high_h);
  z4=(500,latin_wide_low_v);

  push_stroke(z1–z3,(1.6,1.6)–(1.6,1.6));

  push_stroke(z2–z4,(1.6,1.6)–(1.6,1.6));
  set_boserif(0,1,3);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
      ((0,0) transformed accent_default[anc_upper]));
  expand_pbox;
enddef;
```



```
vardef latin.upthorn =
  push_pbox_toexpand("latin.upthorn");
  latin.upp_base(375);
  replace_strokep(0)(oldp
    shifted (-llcorner oldp) yscaled 0.9
    shifted ((llcorner oldp)+(0,vmetric(0)-vmetric(0.18))));
  push_stroke((xpart point infinity of get_strokep(0),latin_wide_high_v)
    –(xpart point infinity of get_strokep(0),latin_wide_low_v),
    (1.6,1.6)–(1.6,1.6));
  set_boserif(0,0,3);
  set_boserif(0,1,3);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
```

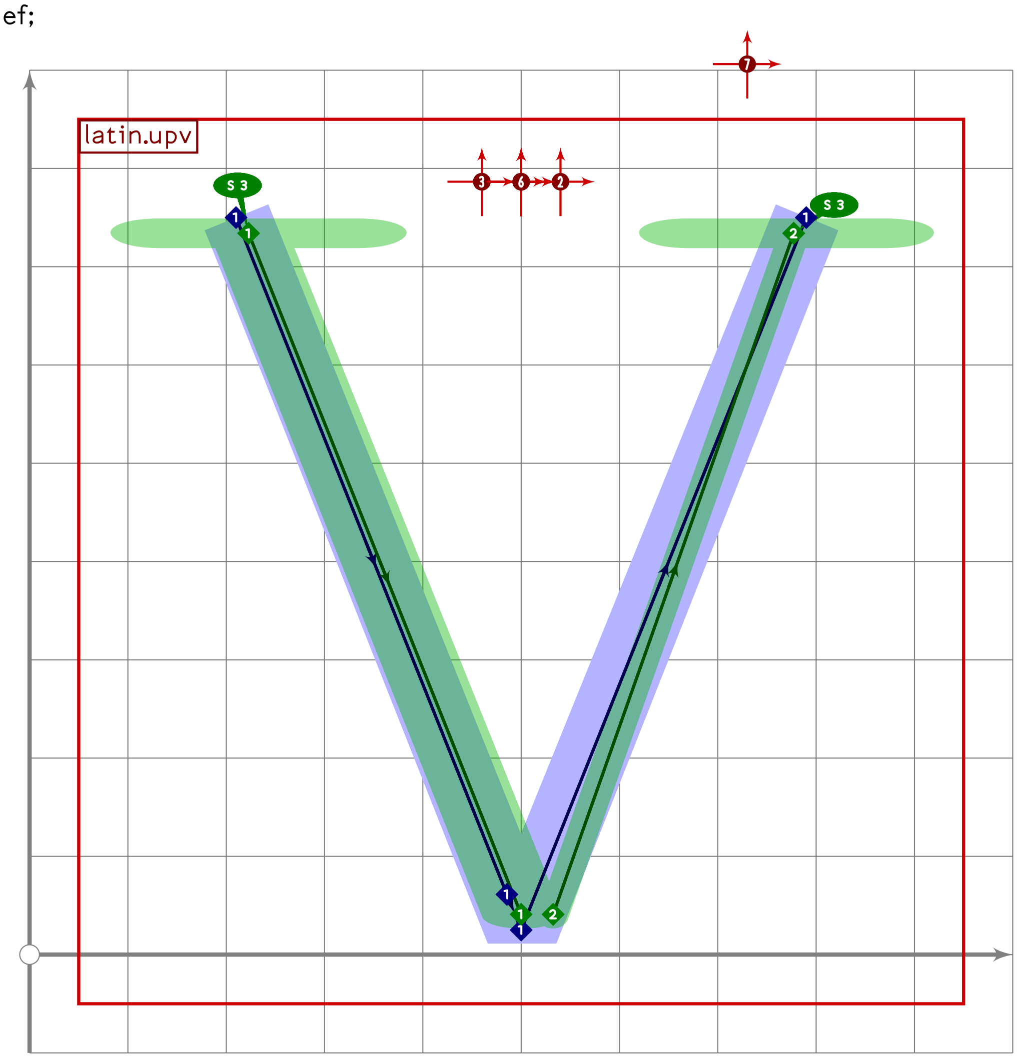
U+FF35
tsuku.uniFF35

```
   expand_pbox;
enddef;
```



```

vardef latin.upu =
   push_pbox_toexpand("latin.upu");
   x1=x2;
   x3=500;
   x4=x5;
   (x1+x5)/2=x3;
   (x5-x1)=(y1-y3)*0.83;

   y1=y5=latin_wide_high_v;
   y2=y4;
   y2-y3=x3-x2;
   y3=latin_wide_low_r;

   push_stroke(z1--z2{dir 274}..z3..{dir 86}z4--z5,
      (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
   set_boserif(0,0,3);
   set_boserif(0,4,3);
```

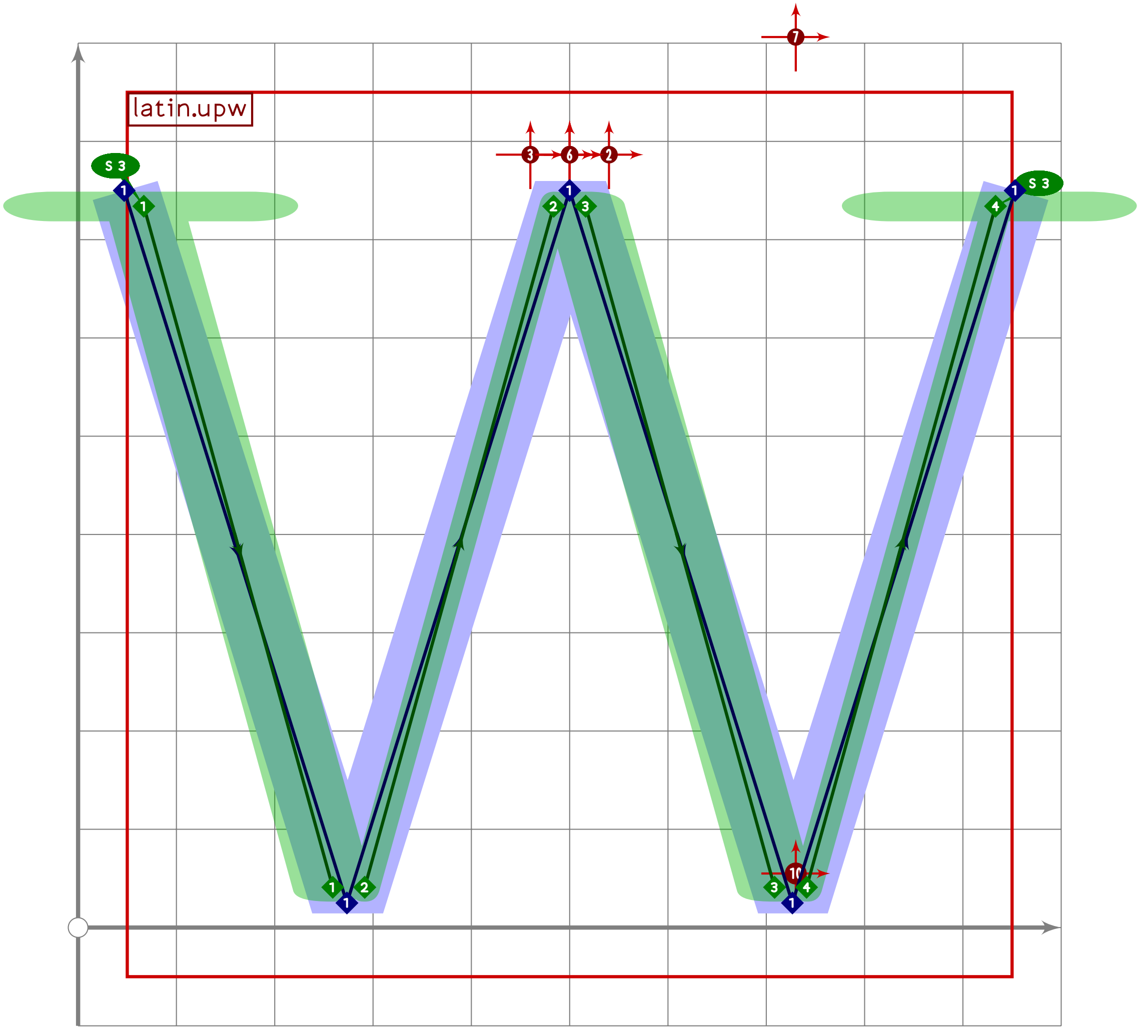
```

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  expand_pbox;
enddef;
```



```

vardef latin.upv =
  push_pbox_toexpand("latin.upv");
  (x1+x3)/2=x2=500;

  y1=y3=latin_wide_high_v;
  y2=latin_wide_low_v;

  (x3-x1)=(y1-y2)*0.8;

  if do_alternation:
    push_stroke(z1--z2,(1.6,1.6)--(1.6,1.6));
    set_boserif(0,0,3);

```

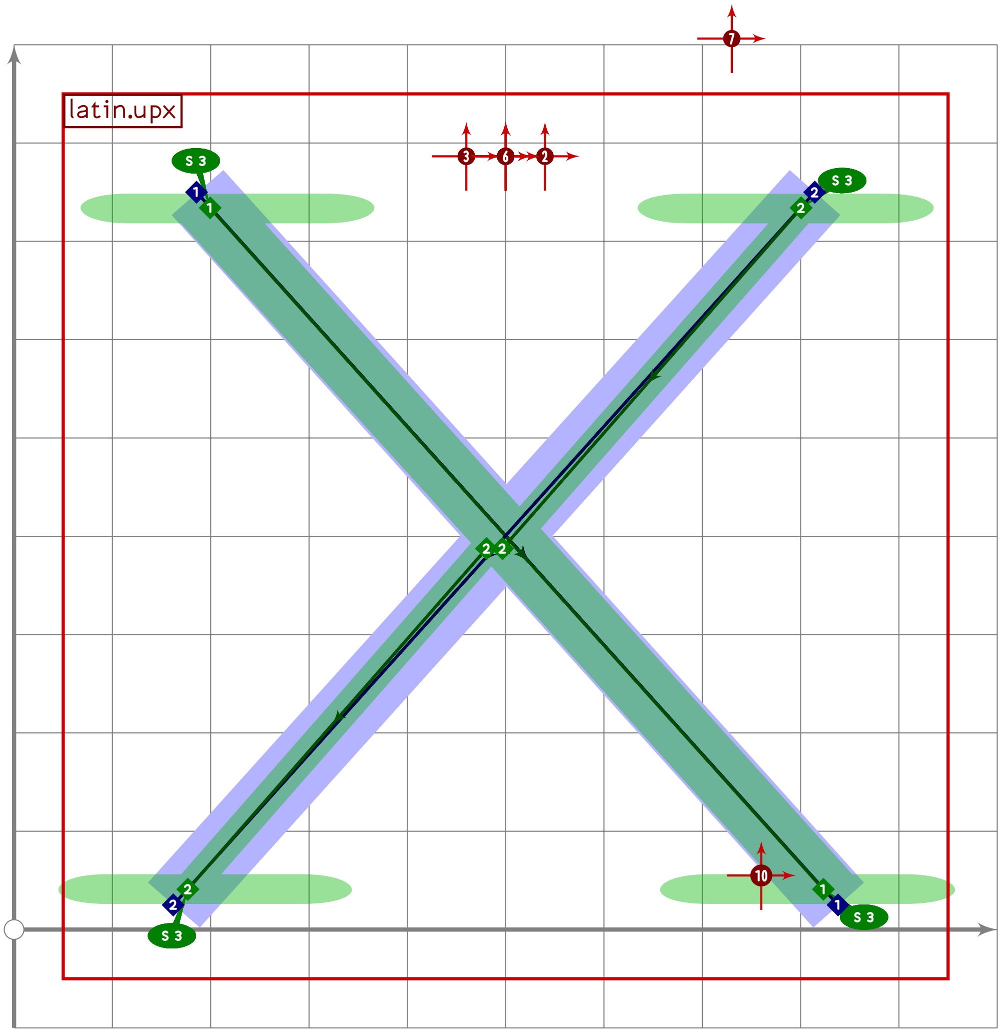
U+FF37
tsuku.uniFF37

```
    push_stroke((z2+alternate_adjust*right)--z3,(1.6,1.6)--(1.6,1.6));
    set_boserif(0,1,3);
    set_boalternate(0);
  else:
    push_stroke(z1--(0.95[z1,z2])--z2--z3,
      (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
    set_botip(0,2,0);
    set_boserif(0,0,3);
    set_boserif(0,3,3);
  fi;

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  expand_pbox;
enddef;
```

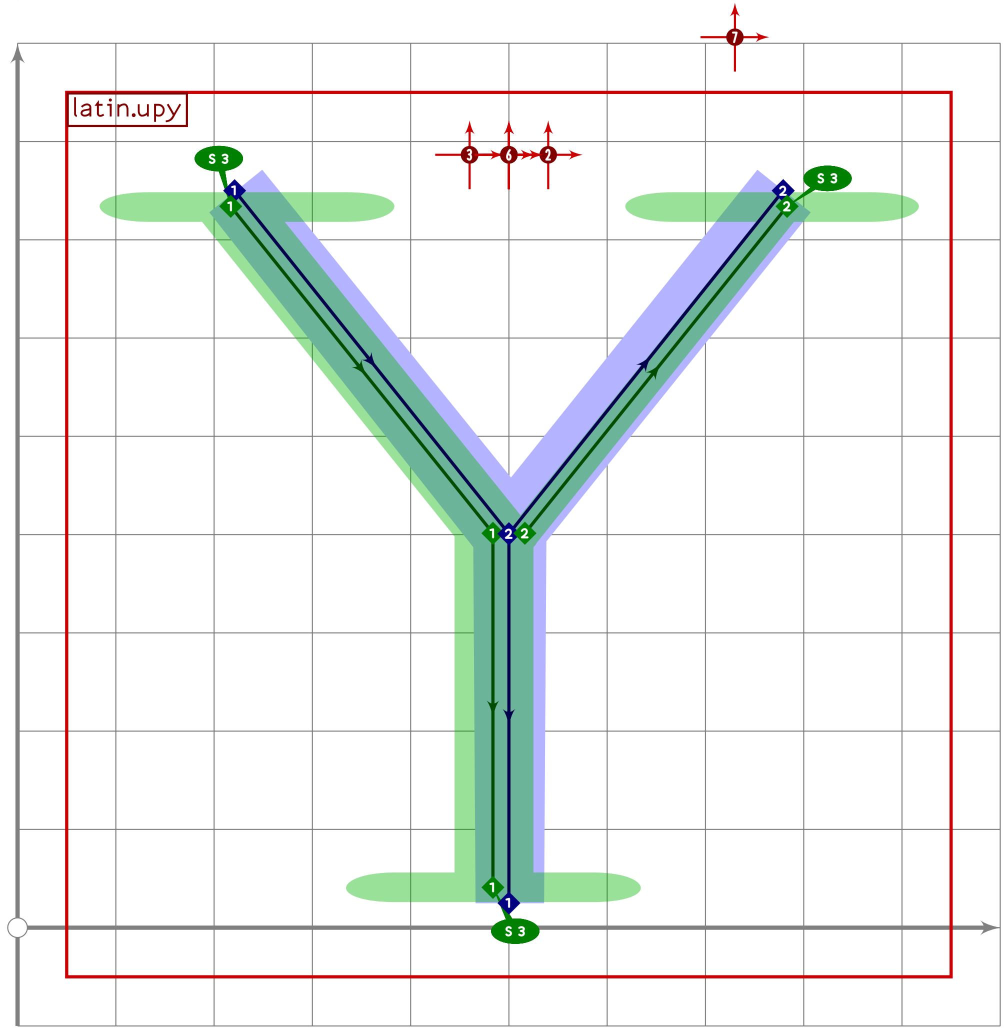


```

vardef latin.upw =
  push_pbox_toexpand("latin.upw");
  if do_alternation:
    (x1+x5)/2=(x2+x4)/2=x3=500-alternate_adjust/2;
    (x3-x2)=(x2-x1);

    y1=y3=y5=latin_wide_high_v;
    y2=y4=latin_wide_low_v;

    (x5-x1)=(y1-y2)*1.25-(3*alternate_adjust);

    push_stroke((z1--z2) shifted (alternate_adjust*left),
      (1.6,1.6)--(1.6,1.6));
    set_boserif(0,0,3);

    push_stroke(z2--z3,(1.6,1.6)--(1.6,1.6));
    set_boalternate(0);

    push_stroke((z3--z4) shifted (alternate_adjust*right),
      (1.6,1.6)--(1.6,1.6));

    push_stroke((z4--z5) shifted (alternate_adjust*right*2),
      (1.6,1.6)--(1.6,1.6));
    set_boserif(0,1,3);
    set_boalternate(0);
  else:
    (x1+x5)/2=(x2+x4)/2=x3=500;
    (x3-x2)=(x2-x1);

    y1=y3=y5=latin_wide_high_v;
    y2=y4=latin_wide_low_v;

    (x5-x1)=(y1-y2)*1.25;

    push_stroke(z1--z2--z3--z4--z5,
      (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
    set_botip(0,1,0);
    set_botip(0,2,0);
    set_botip(0,3,0);
    set_boserif(0,0,3);
    set_boserif(0,4,3);
  fi;

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  tsu_accent.shift_anchors(ai=anc_lower_connect)((230,0));
  expand_pbox;
enddef;
```

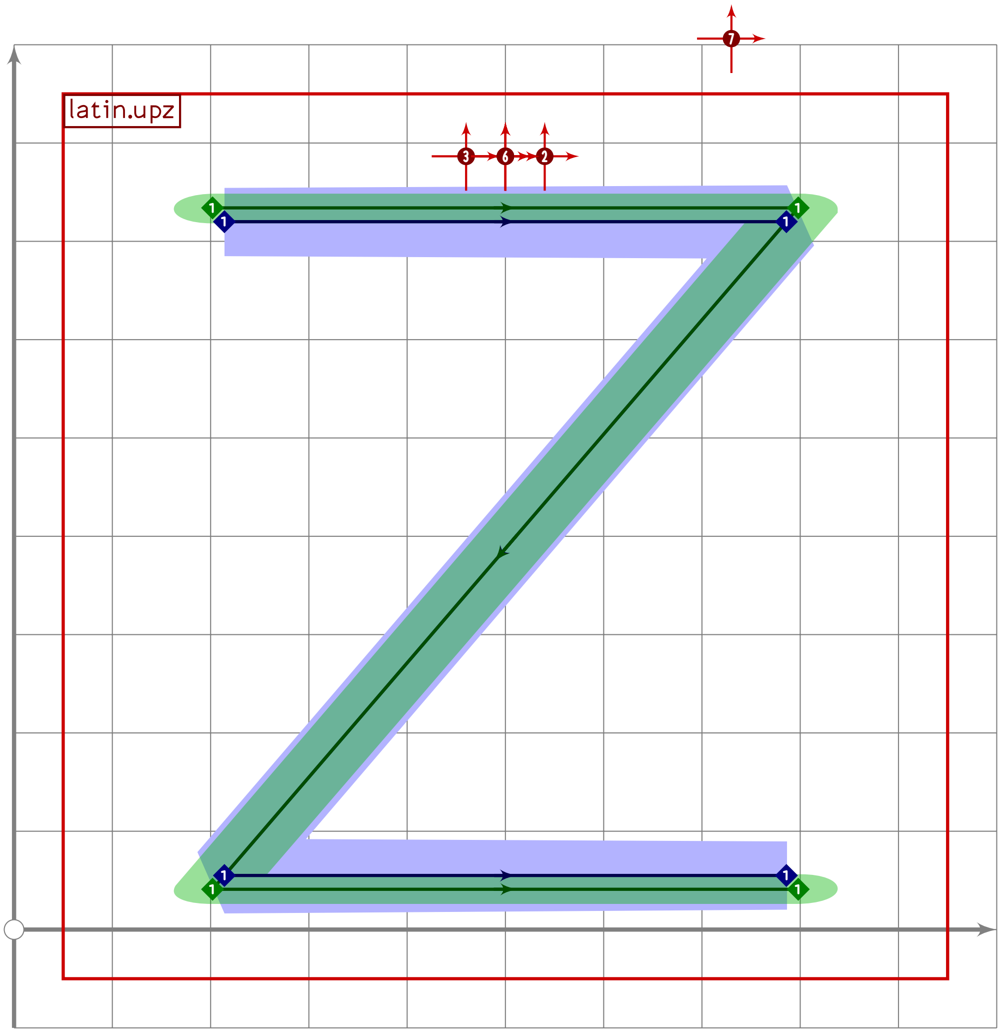
U+FF38
tsuku.uniFF38



```

vardef latin.upx =
  push_pbox_toexpand("latin.upx");
  (x1+x3)/2=500;
  (x2+x4)/2=500;
  (x2+x3-x1-x4)=((y1-y2)*0.9)*2;
  (x3-x1)=(x2-x4)*0.93;

  y1=y3=latin_wide_high_v;
  y2=y4=latin_wide_low_v;

  push_stroke(z1--z2,(1.6,1.6)--(1.6,1.6));
  set_boserif(0,0,3);
  set_boserif(0,1,3);

  if do_alternation:
    push_stroke(z3--(0.5[z3,z4]+alternate_adjust*right/4)
      --(0.5[z3,z4]+alternate_adjust*left/4)--z4,
      (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
    set_boserif(0,0,3);
    set_boserif(0,3,3);
  else:
    push_stroke(z3--z4,(1.6,1.6)--(1.6,1.6));
    set_boserif(0,0,3);
    set_boserif(0,1,3);
  fi;
  set_boalternate(0);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  tsu_accent.shift_anchors(ai=anc_lower_connect)((260,0));
  expand_pbox;
enddef;
```

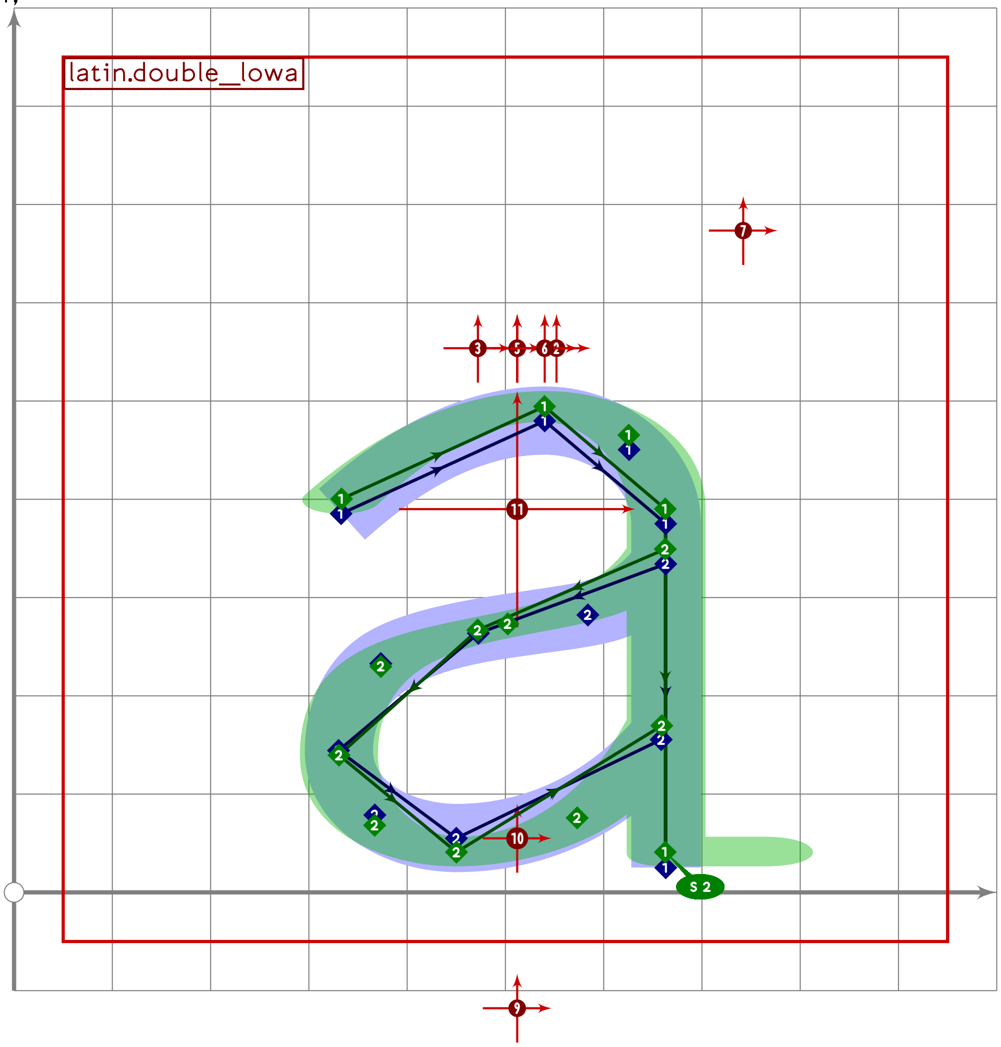


```

vardef latin.upy =
  push_pbox_toexpand("latin.upy");
  (x3+x1)/2=x2=x4=if do_alternation: 500-alternate_adjust/2 else: 500 fi;
  (x3-x1)=0.77*(y1-y4);

  y1=y3=latin_wide_high_v;
  y2=vmetric(0.52);
  y4=latin_wide_low_v;

  push_stroke(z1--z2--z4,(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,1,0);
  set_boserif(0,0,3);
  set_boserif(0,2,3);

  if do_alternation:
    push_stroke((z2--z3) shifted (alternate_adjust*right),
      (1.6,1.6)--(1.6,1.6));
  else:
    push_stroke(z2--z3,(1.6,1.6)--(1.6,1.6));
  fi;
  set_boserif(0,1,3);
  set_boalternate(0);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  expand_pbox;
enddef;
```



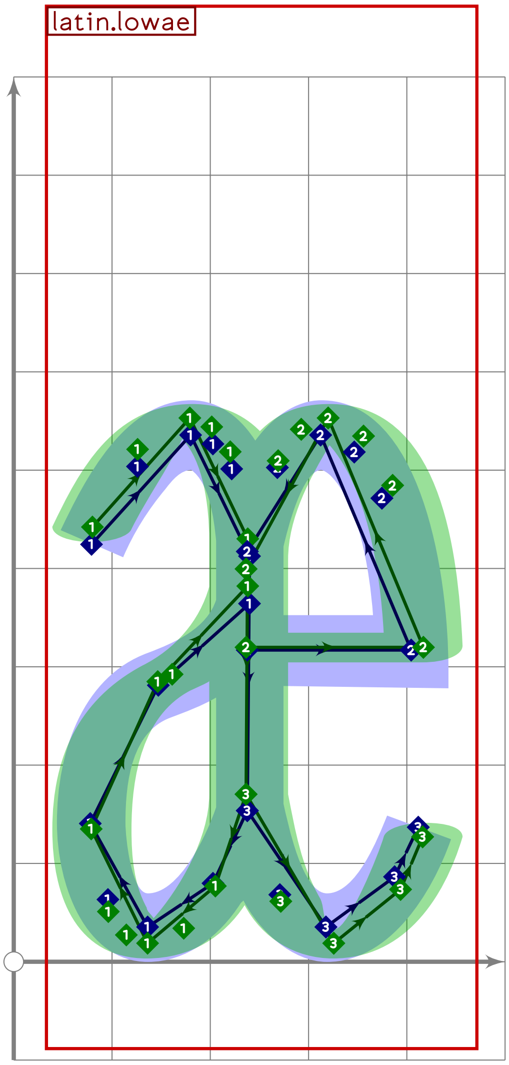
```
vardef latin.upz =
  push_pbox_toexpand("latin.upz");
  y1=y2=latin_wide_high_h;
  y3=y4=latin_wide_low_h;

  x1=x3;
  x2=x4;
  (x1+x2)/2=500;
  (x2-x1)=(y1-y3)*0.86;

  push_stroke(z1--z2--z3--z4,(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,1,0);
  set_botip(0,2,0);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  expand_pbox;
enddef;
```

```


vardef latin.double_lowa =
  push_pbox_toexpand(”latin.double_lowa”);
  x1=(-0.12)[x6,x3];
  x3=x4=x5=x8;
  0.51[x6,x3]=500;
  x3-x1=0.82*(y2-y4);
  x2=0.63[x6,x3];
  x7=0.36[x6,x3];

  y1=0.7[y4,y2];
  y3=0.77[y4,y2];
  y2=latin_wide_xheight_r;
  y4=latin_wide_low_v;
  y5=0.68[y4,y2];
  y6=0.32[y7,y5];
  y7=latin_wide_low_h;
  y8=0.3[y4,y2];

  push_stroke(z1{curl 0.2}..z2{right}..z3{down}..z4,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(subpath (0.2,3) of oldp);
  set_boserif(0,3,2);

  push_stroke(z5{dir 240}..z6{down}..z7{right}..z8,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(subpath (0,2.97) of oldp);
  replace_strokep(0)(insert_nodes(oldp)(0.5));

  tsu_accent.shift_anchors(true)((12,0));
  tsu_accent.shift_anchors(ai=anc_ring)((28,0));
  expand_pbox;
enddef;

vardef latin.single_lowa =
  push_pbox_toexpand(”latin.single_lowa”);
  (x1+x4)/2=520;
  (x1-x4)=(y3-y1)*0.81;
  x1=x6-8;
  x3=0.4[x1,x4];
  x5=0.36[x4,x1];

  y1=latin_wide_low_v;
  y3=latin_wide_xheight_h;
  y4=0.47[y5,y3];
  y5=latin_wide_low_h;
  y6=0.37[y1,y3];

  z7=(x1,y3);
  z2=0.3[z7,0.5[z3,z1]];

  push_stroke(interpath(0.5)
      (z1{up}..{up}z7{left}..{left}z3..{down}z4..{right}z5..z6,
       z1{up}..z2..{left}z3..{down}z4..{right}z5..z6),
    (1.3,1.3)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--
      (1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(reverse(insert_nodes(oldp)(0.5)));
  set_boserif(0,6,if do_italic_hook: 11 else: 2 fi);
  set_botip(0,4,1);
  expand_pbox;
enddef;
```



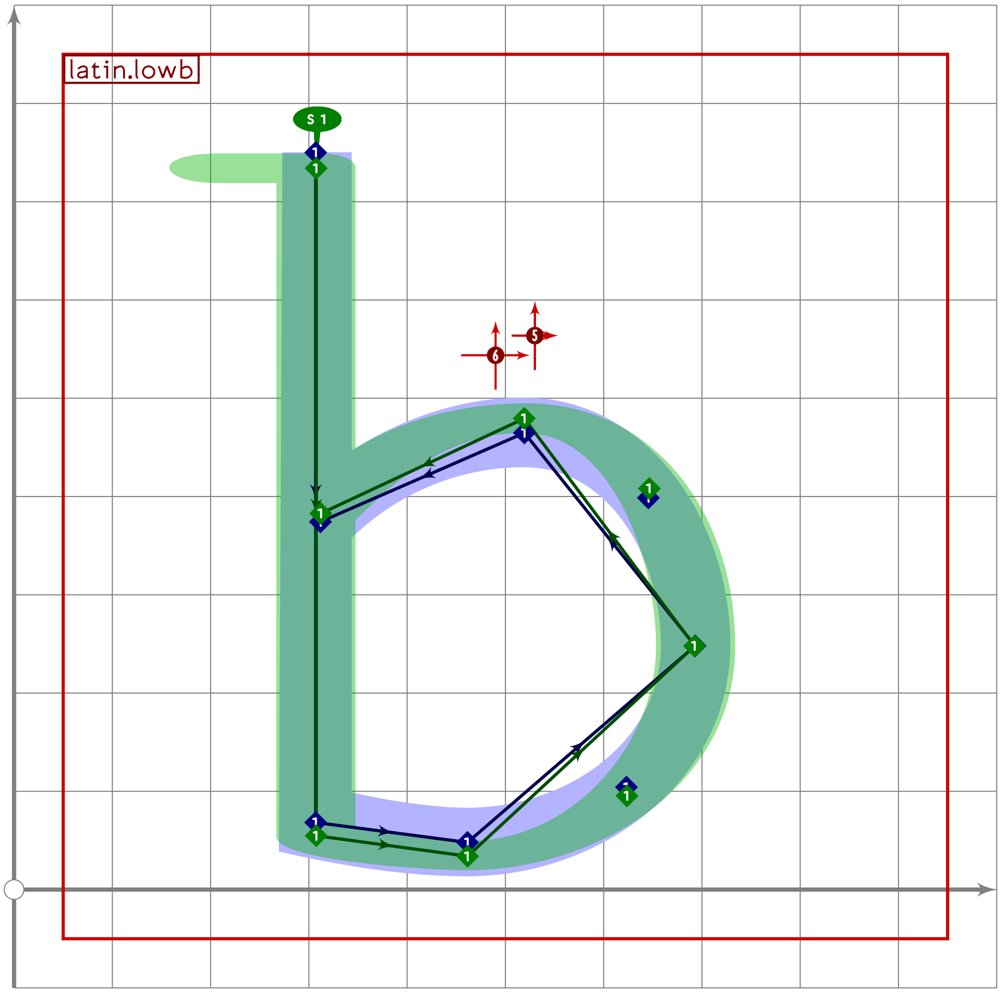
```

vardef latin.lowa =
  latin.double_lowa;
enddef;
```

U+00E6
tsuku.ae

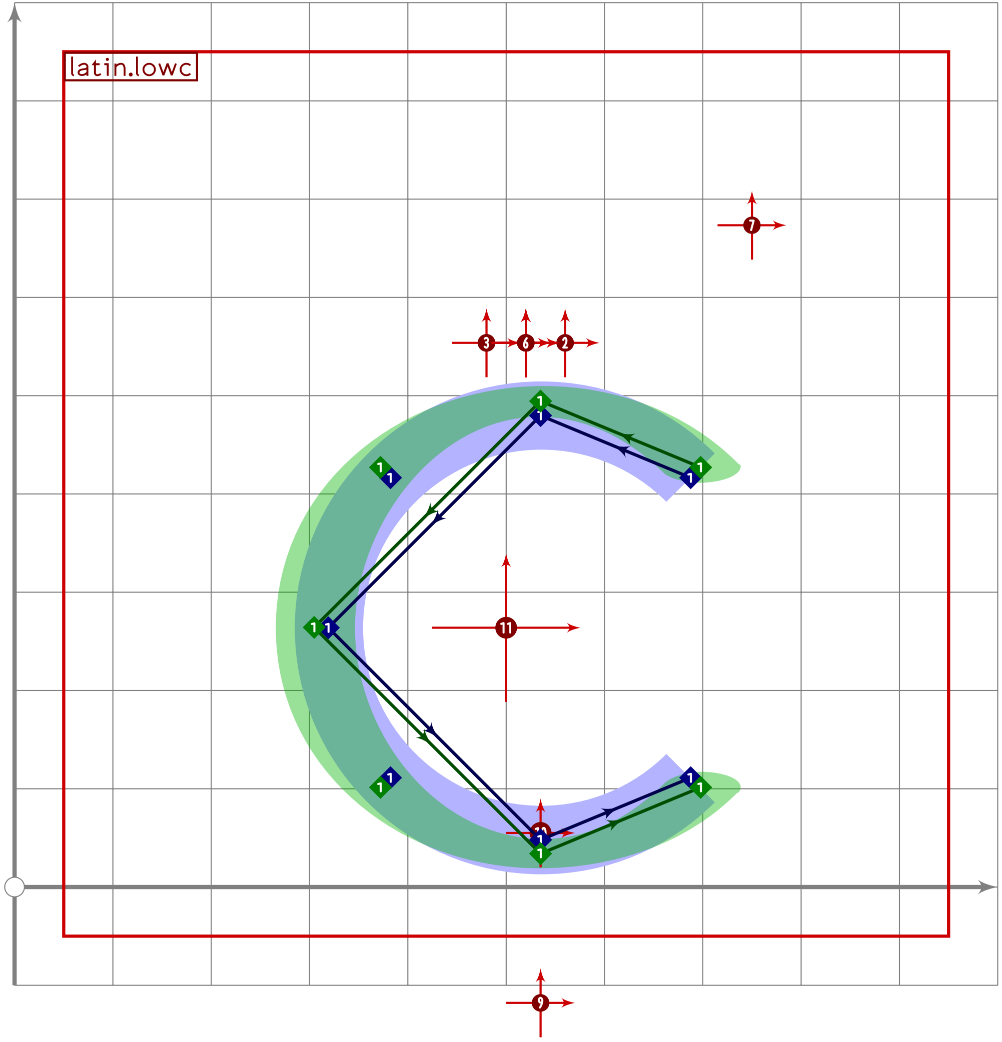


```

vardef latin.lowae =
  push_pbox_toexpand("latin.lowae");
  begingroup
    save saved_sp;
    save astem,abowl,astroke,estroke,etop,ebot;
    path astem,abowl,astroke,estroke,etop,ebot;
    saved_sp:=sp;

    latin.double_lowa;
    set_boserif(-1,3,whatever);
    astem:=get_strokep(-1);
    abowl:=get_strokep(0);

    numeric x[],y[];
    latin.lowe;
    estroke:=get_strokep(0);

    for i=saved_sp upto sp-1:
      obstacktype[i]:=otnull;
    endfor;

    astroke:=(subpath (0,2) of astem)--reverse abowl;

    numeric x[],y[];
    z1=point 3 of astroke;
    path xbp;
    xbp=estroke shifted (0,ypart ((llcorner astroke)-(llcorner estroke)));
    x2=xpart (xbp intersectionpoint ((470,y1)--(0,y1)));
    astroke:=astroke shifted (470-x1,0);

    estroke:=estroke shifted (470-x2,0);
    xbp:=xbp shifted (470-x2,0);

    ebot:=subpath (xpart (xbp intersectiontimes ((500,y1)--(0,y1))),
                 infinity) of xbp;
    ebot:=insert_nodes(ebot)((length ebot)-0.3);

    etop:=
      subpath (ypart (((470,1000)--(470,0))
          intersectiontimes (subpath (0,1) of estroke)),
        1+ypart (((470,1000)--(470,0))
          intersectiontimes (subpath (1,infinity) of estroke)))
        of estroke;

    astroke:=insert_nodes(astroke)(3.4);

    push_stroke(astroke,
      (1.6,1.6) for i=1 upto length astroke: --(1.6,1.6) endfor);
    push_stroke(etop,
      (1.6,1.6) for i=1 upto length etop: --(1.6,1.6) endfor);
    set_botip(0,1,1);
    push_stroke(ebot,
      (1.6,1.6) for i=1 upto length ebot: --(1.6,1.6) endfor);
  endgroup;
  expand_pbox;
enddef;
```

U+FF42
tsuku.uniFF42



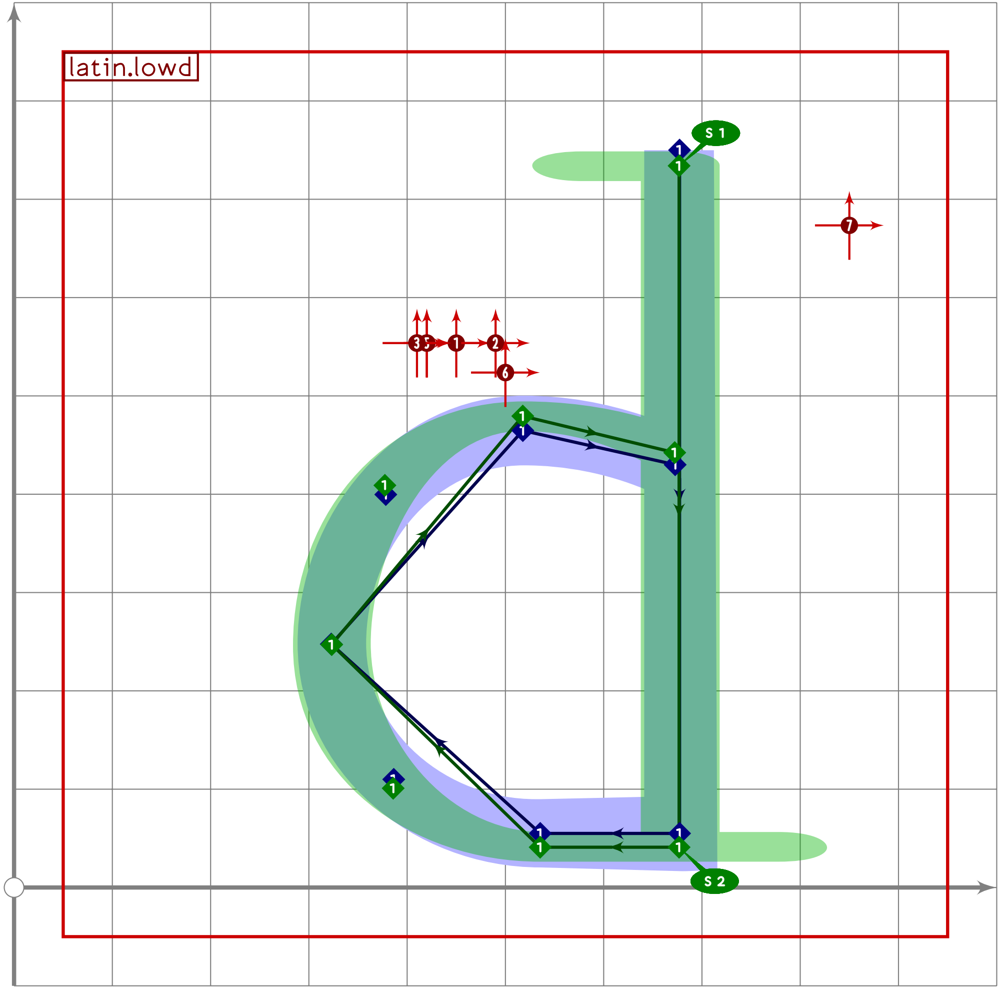
```

vardef latin.lowb =
  push_pbox_toexpand("latin.lowb");
  (x1+x4)/2=500;
  (x4-x1)=(y1-y3)*0.55;
  x2=x1=x6;
  x3=0.4[x2,x4];
  x5=0.55[x2,x4];

  y1=latin_wide_high_v;
  y2=latin_wide_lc_baselift;
  y3=latin_wide_low_r;
  y4=0.48[y3,y5];
  y5=latin_wide_xheight_h;
  y6=0.77[y3,y5];

  push_stroke(z1--z2{curl 0.05}..{right}z3..{up}z4..{left}z5..z6,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(subpath (0,4.97) of oldp);
  set_botip(0,1,1);
  if not do_italic_hook: set_boserif(0,0,1); fi;
  push_anchor(anc_wide,identity xscaled 0.9
    transformed accent_default[anc_wide] shifted (30,10));
  push_anchor(anc_tilde,identity xscaled 0.8
    transformed accent_default[anc_tilde] shifted (30,10));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (-10,-10));
  expand_pbox;
enddef;
```



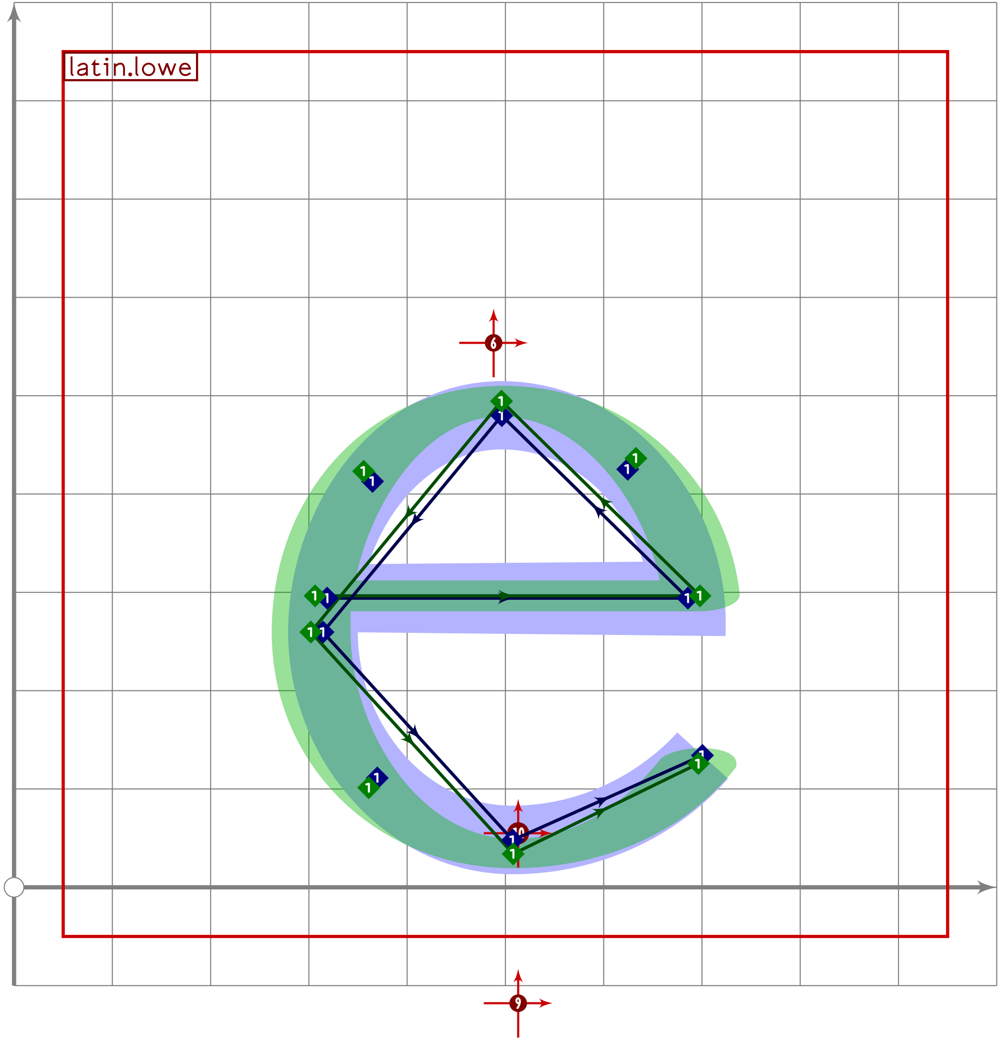
```
vardef latin.lowc =
  push_pbox_toexpand(”latin.lowc”);
  push_stroke((subpath (0.5,3.5) of ((1,0)..(0,1)..(-1,0)..(0,-1)..cycle))
    scaled ((latin_wide_xheight_r-latin_wide_low_r)/2)
    shifted ((xpart centre_pt,(latin_wide_xheight_r+latin_wide_low_r)/2)
      +(35,0)),
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))((20,0));
  tsu_accent.shift_anchors(ypart olda<=vmetric(0.52))((35,0));
  push_anchor(anc_centre,identity
    scaled ((latin_wide_xheight_r-latin_wide_low_r)/200)
```

U+FF44
tsuku.uniFF44

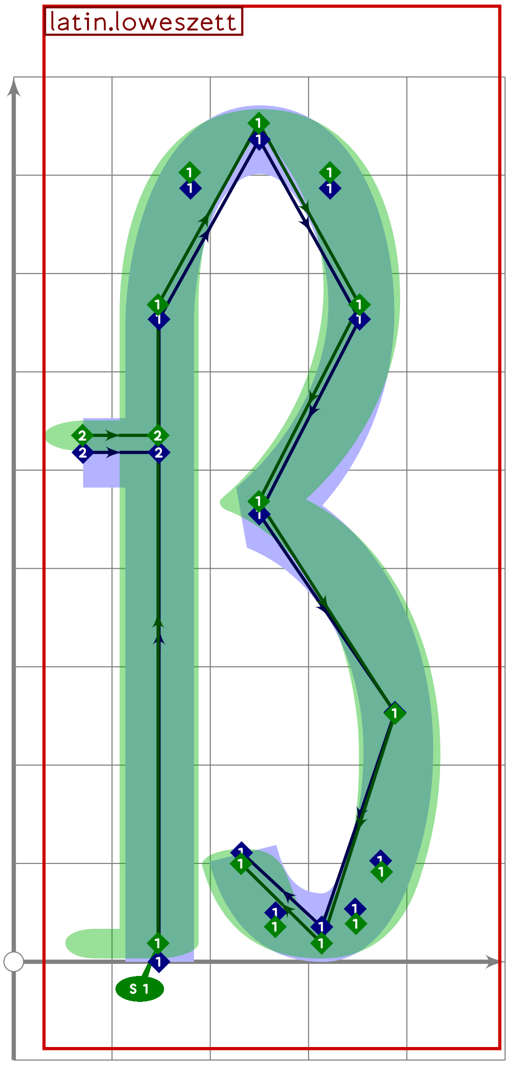
```
    shifted (xpart centre_pt,(latin_wide_xheight_r+latin_wide_low_r)/2));
  expand_pbox;
enddef;
```



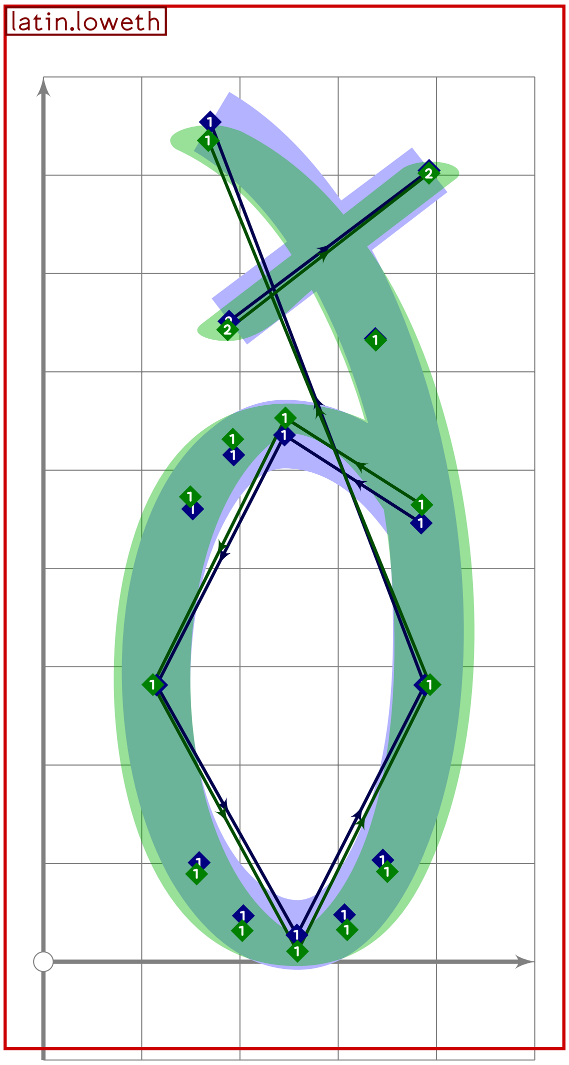
```

vardef latin.lowd =
  push_pbox_toexpand("latin.lowd");
  (x1+x4)/2=500;
  (x1-x4)=(y1-y3)*0.51;
  x2=x1=x6;
  x3=0.4[x2,x4];
  x5=0.45[x2,x4];

  y1=latin_wide_high_v;
  y2=y3=latin_wide_low_h;
  y4=0.47[y3,y5];
  y5=latin_wide_xheight_h;
  y6=0.91[y3,y5];

  push_stroke(z1--z2{left}..{left}z3..{up}z4..{right}z5..z6,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--
      (1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(subpath (0,4.97) of oldp);
  set_botip(0,1,1);
  if not do_italic_hook: set_boserif(0,0,1); fi;
  set_boserif(0,1,2);

  push_anchor(anc_wide,identity xscaled 0.9
    transformed accent_default[anc_wide] shifted (-30,0));
  push_anchor(anc_tilde,identity xscaled 0.8
    transformed accent_default[anc_tilde] shifted (-30,0));

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))((-50,0));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (0,-30));
  push_anchor(anc_caron_comma,accent_default[anc_caron_comma] shifted (120,0));
  expand_pbox;
enddef;
```



```

vardef latin.lowe =
  push_pbox_toexpand("latin.lowe");
  y2=0.57[y5,y3];
  y3=latin_wide_xheight_r;
  y4=0.49[y5,y3];
  y5=latin_wide_low_r;
  y6=0.35[y5,y2];

  (x2+x4)/2=500;
  (x2-x4)=0.86*(y3-y5);
  x3=0.49[x4,x2];
  x5=0.52[x4,x2];
  x6=1.04[x4,x2]-(if sharp_corners: 0 else: (mbrush_width/3) fi);

  push_stroke(z2{curl 0.7}..z3{left}..z4{down}..z5{right}..z6,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  z1=get_strokep(0) intersectionpoint ((z2+(-1000,0))--z2+(-10,0));
  replace_strokep(0)((z1+2*right)--oldp);
  set_botip(0,1,1);

  tsu_accent.shift_anchors(ypart olda<vmetric(0.05))((13,0));
  tsu_accent.shift_anchors(ai=anc_ring)((-12,0));
  expand_pbox;
enddef;
```

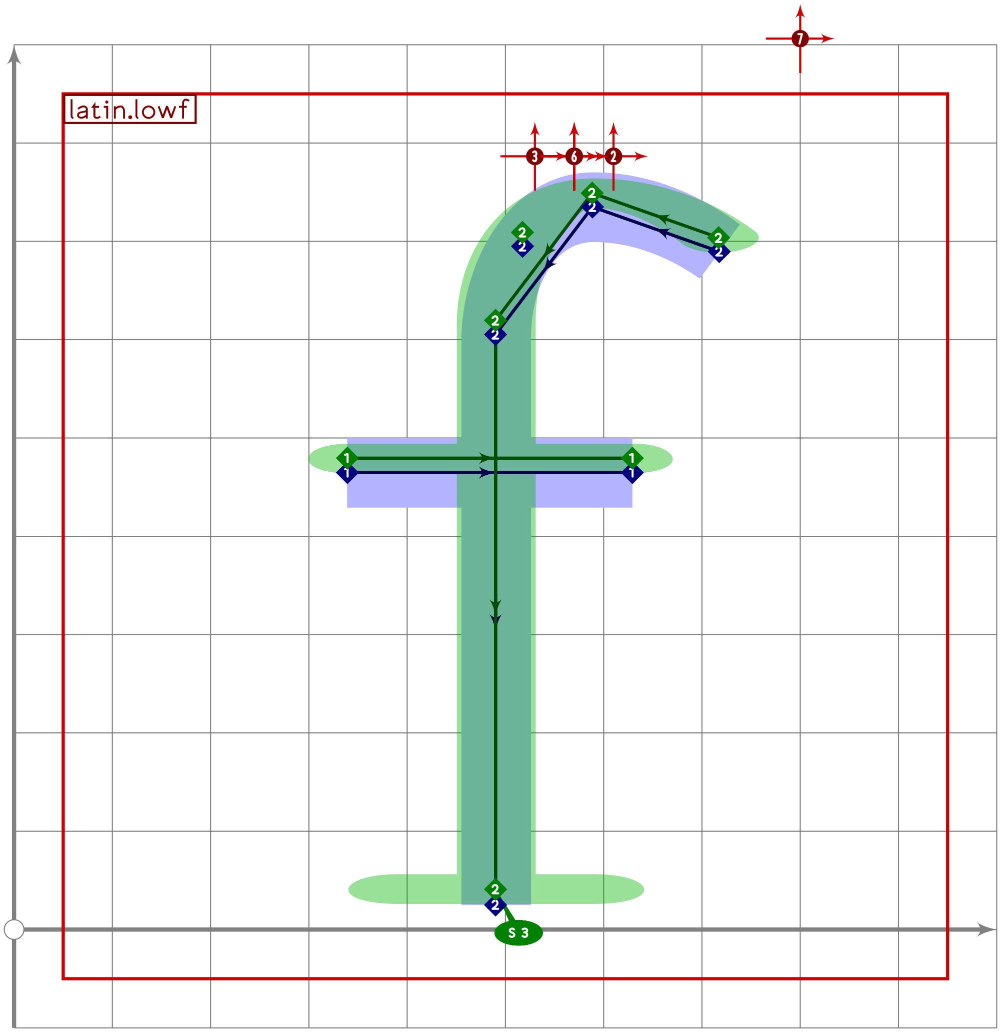


```

vardef latin.loweszett =
  push_pbox_toexpand("latin.loweszett");
  (x1+x6)/2=510;
  (x6-x1)=3*(y3-y2);
  x2=x1;
  x3=x5=(x1+x4)/2;
  x4=0.85[x1,x6];
  x7=0.69[x1,x6];
  x8=0.35[x1,x6];

  y1=latin_wide_low_v;
  y2=y4=0.52[y5,y3];
  y3=latin_wide_high_r;
  y5=0.87[y7,latin_wide_xheight_h];
  y6=0.52[y7,y5];
  y7=latin_wide_low_h;
  y8=0.18[y7,y5];

  push_stroke(z1--z2{dir 88}..z3{right}..z4..{dir 200}z5{dir 350}..
      z6..z7{left}..z8{dir 120},
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--
      (1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,4,0);
  set_boserif(0,0,1);

  push_stroke((x1-150,latin_wide_xheight_h)--(x1,latin_wide_xheight_h),
    (1.6,1.6)--(1.6,1.6));
  expand_pbox;
enddef;
```

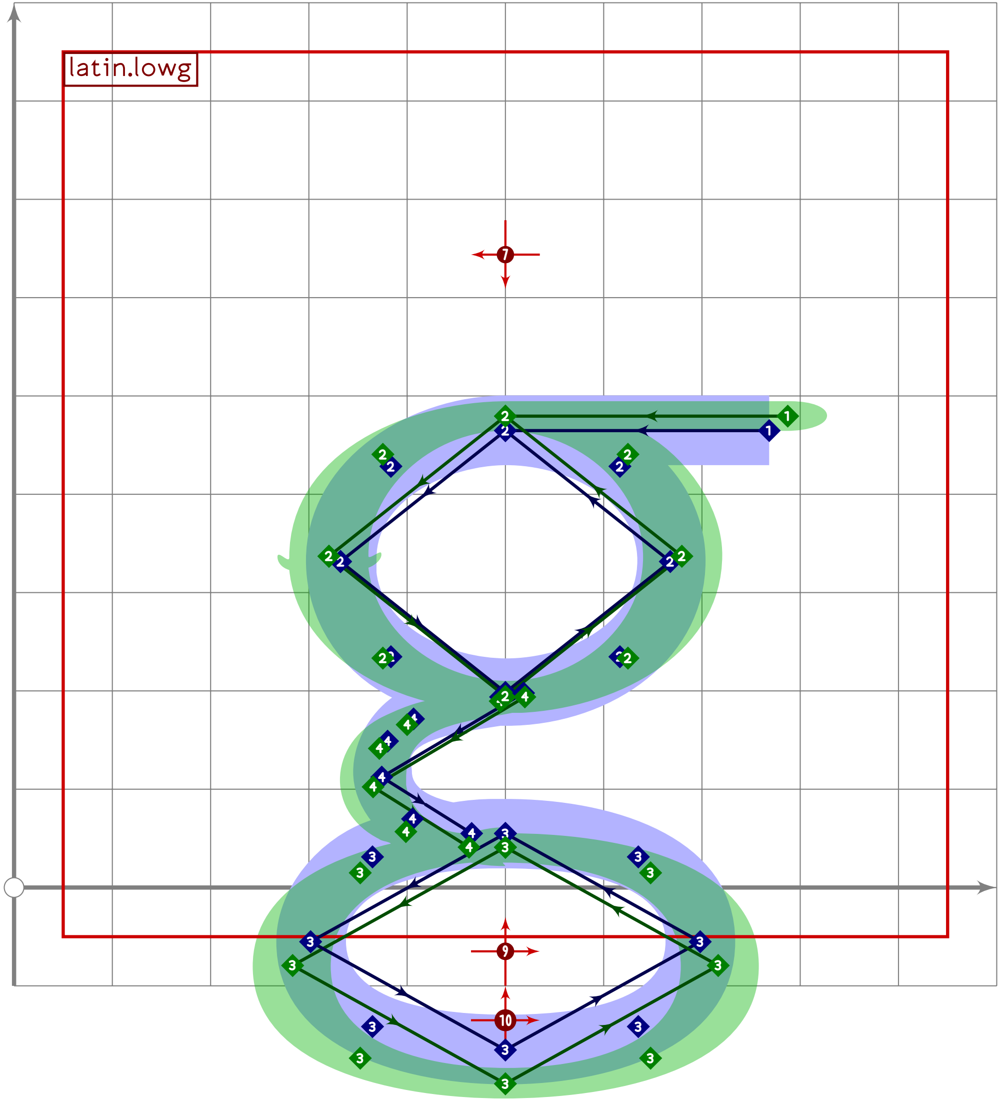


```

vardef latin.loweth =
  push_pbox_toexpand(”latin.loweth”);
  (x3+x5)/2=(x2+x4)/2=510;
  x4-x2=20;
  (x5-x3)=(y2-y4);
  x1=1.3[x3,x5];
  x6=0.2[x3,x5];

  y1=0.25[y4,y2];
  y3=y5=0.5[y4,y2];
  y2=latin_wide_xheight_r;
  y4=latin_wide_low_r;
  y6=latin_wide_high_v;

  push_stroke(z1..z2{left}..z3..z4{right}..z5..{curl 0.6}z6,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(subpath (xpart ((subpath (0,1) of oldp)
    intersectiontimes
      (subpath (2,infinity) of oldp)),infinity) of oldp);

  z7=point 4.67 of get_strokep(0);
  z8=z7+whatever*dir 202;
  x8=0.27[x3,x5];
  z9=2[z8,z7];

  push_stroke(z8--z9,(1.5,1.5)--(1.5,1.5));
  set_bosize(0)(85);
  expand_pbox;
enddef;
```

U+FF46
tsuku.uniFF46

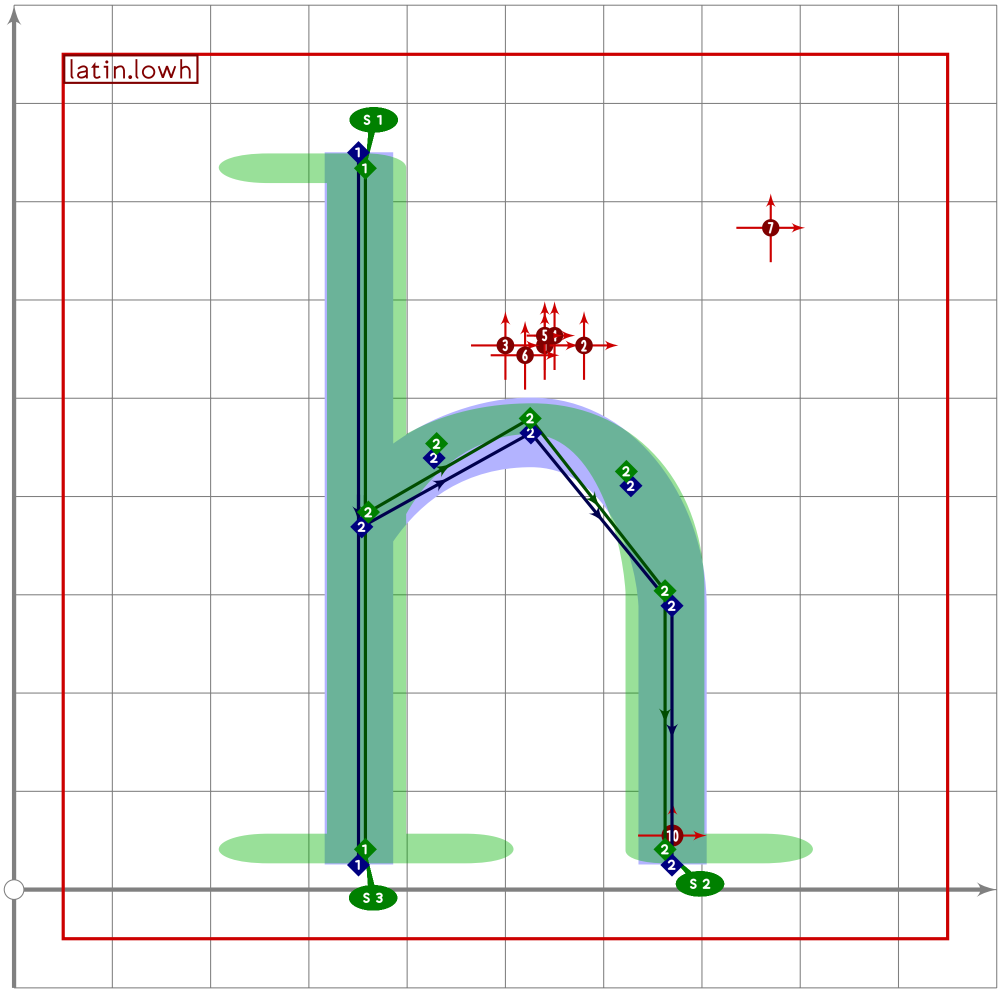


```
vardef latin.lowf =
  push_pbox_toexpand("latin.lowf");
  (x2-x1)=290;
  x5=x6=490=0.52[x1,x2];
  x3-x5=2*(y4-y5);
  x4=0.38[x5,x3];

  y1=y2=latin_wide_xheight_h;
  y5=0.52[y2,y4];
  y3=0.73[y2,y4];
  y4=latin_wide_high_r;
  y6=latin_wide_low_v;

  push_stroke(z1--z2,(1.6,1.6)--(1.6,1.6));

  push_stroke(z3{curl 0.6}..z4{left}..{dir 268}z5{down}--z6,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(subpath (0.23,3) of oldp);
  set_boserif(0,3,3);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper])+(70,0));
  expand_pbox;
enddef;
```



```

vardef latin.lowg =
  push_pbox_toexpand(”latin.lowg”);
  x2=x4=x7=x9=500;
  x1=1.6[x2,x5];
  x5-x2=x2-x3;
  x5-x3=1.26*(y2-y4);
  x6=0.25[x3,x2];
  x10-x7=x7-x8;
  x10-x8=1.8*(y7-y9);

  y1=y2=latin_wide_xheight_h;
  y3=y5=0.5[y4,y2];
  y4=0.35[y7,y2];
  y6=0.4[y7,y4];
  y7=latin_wide_low_h;
  y8=y10=0.5[y9,y7];
  y9=latin_wide_desc_r;

  push_stroke(z1--z2,(1.6,1.6)--(1.6,1.6));

  push_stroke(z3..z4..z5..z2..cycle,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--cycle);

  push_stroke(z7..z8..z9..z10..cycle,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--cycle);

  push_stroke((point 1.2 of get_strokep(-1))
      {-direction 1.2 of get_strokep(-1)}..z6..
      (point 0.05 of get_strokep(0)){-direction 0.05 of get_strokep(0)},
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(subpath (0.13,1.87) of oldp);
  set_bosize(0,85);

  push_anchor(anc_caron_comma,
    identity rotated 180 shifted (0,90)
    transformed accent_default[anc_upper]);
  push_anchor(anc_lower,
    accent_default[anc_lower] shifted (0,53));
  push_anchor(anc_lower_connect,
    accent_default[anc_lower_connect] shifted (0,-190));
  expand_pbox;
enddef;
```

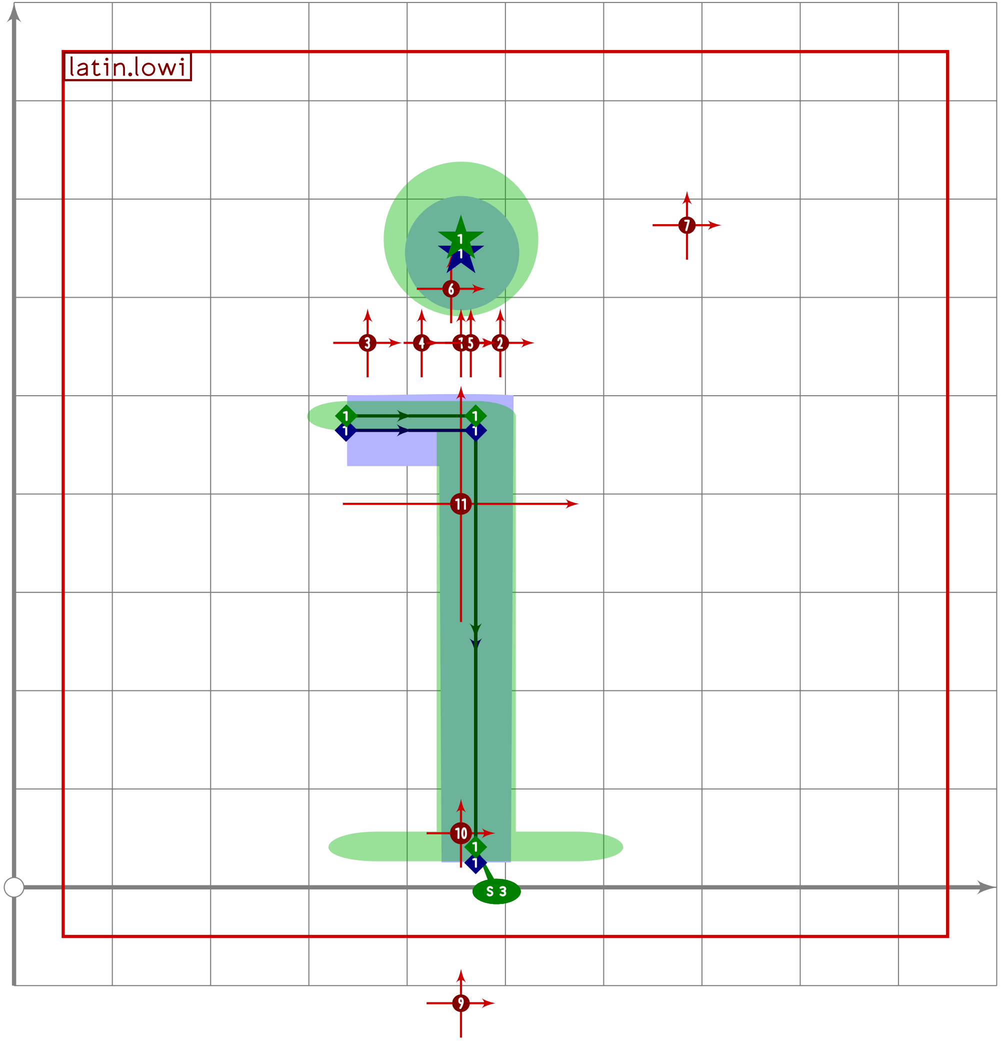
U+FF48
tsuku.uniFF48



```
vardef latin.lowh =
  push_pbox_toexpand(”latin.lowh”);
  (x1+x5)/2=510;
  (x5-x1)=(y1-y2)*0.44;
  x2=x1=x3;
  x4=0.55[x3,x5];
  x6=x5;

  y1=latin_wide_high_v;
  y2=y6=latin_wide_low_v;
  y3=0.77[y2,y4];
  y4=latin_wide_xheight_h;
  y5=0.60[y2,y4];

  push_stroke(z1--z2,(1.6,1.6)--(1.6,1.6));
  if not do_italic_hook:
    set_boserif(0,0,1);
    set_boserif(0,1,3);
  fi;

```

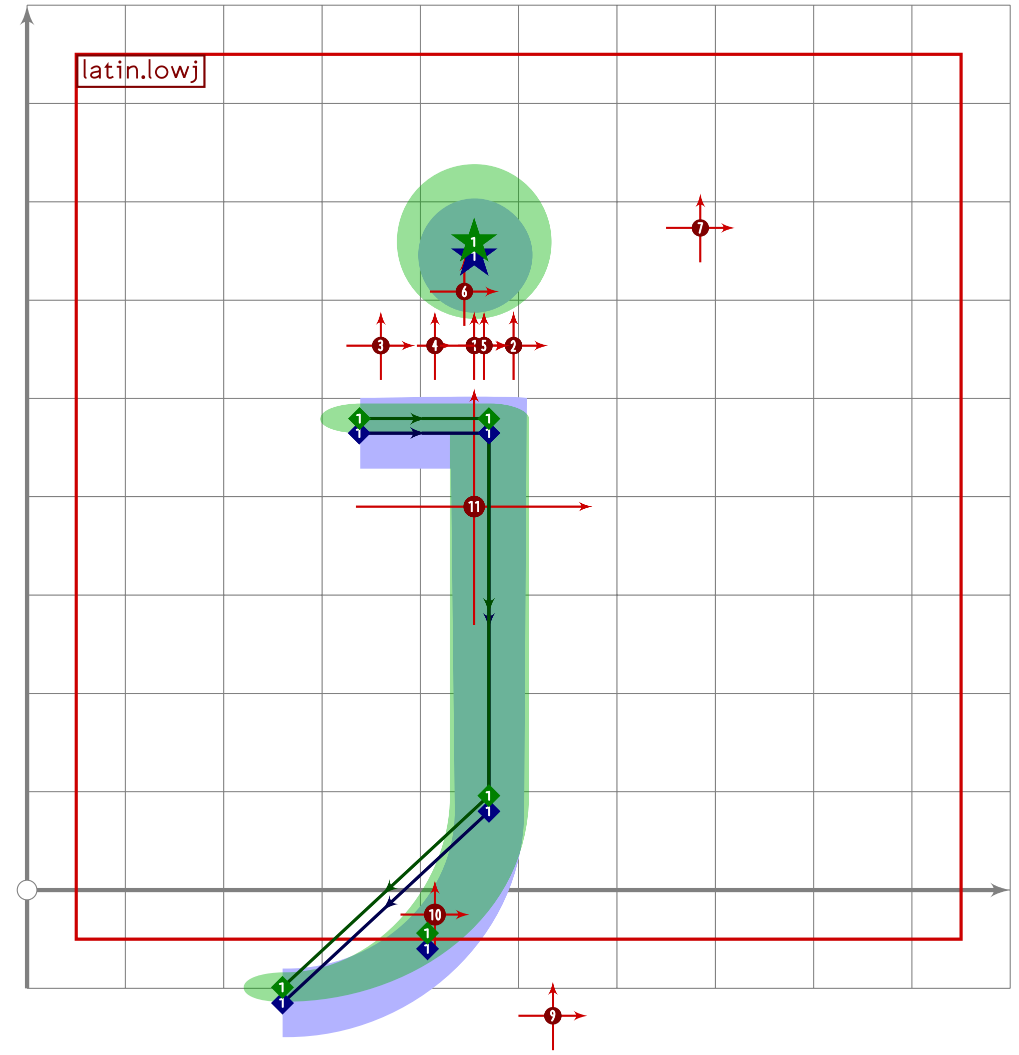
U+FF49
tsuku.uniFF49

```
  push_stroke(z3..z4{right}..z5{dir 273}--z6,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(subpath (0.03,3) of oldp);
  set_boserif(0,3,if do_italic_hook: 11 else: 2 fi);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))((40,0));
  push_anchor(anc_wide,identity xscaled 0.8
    transformed accent_default[anc_wide] shifted (50,10));
  push_anchor(anc_tilde,identity xscaled 0.7
    transformed accent_default[anc_tilde] shifted (40,10));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (20,-10));
  push_anchor(anc_lower_connect,
    accent_default[anc_lower_connect] shifted (170,0));
  expand_pbox;
enddef;
```

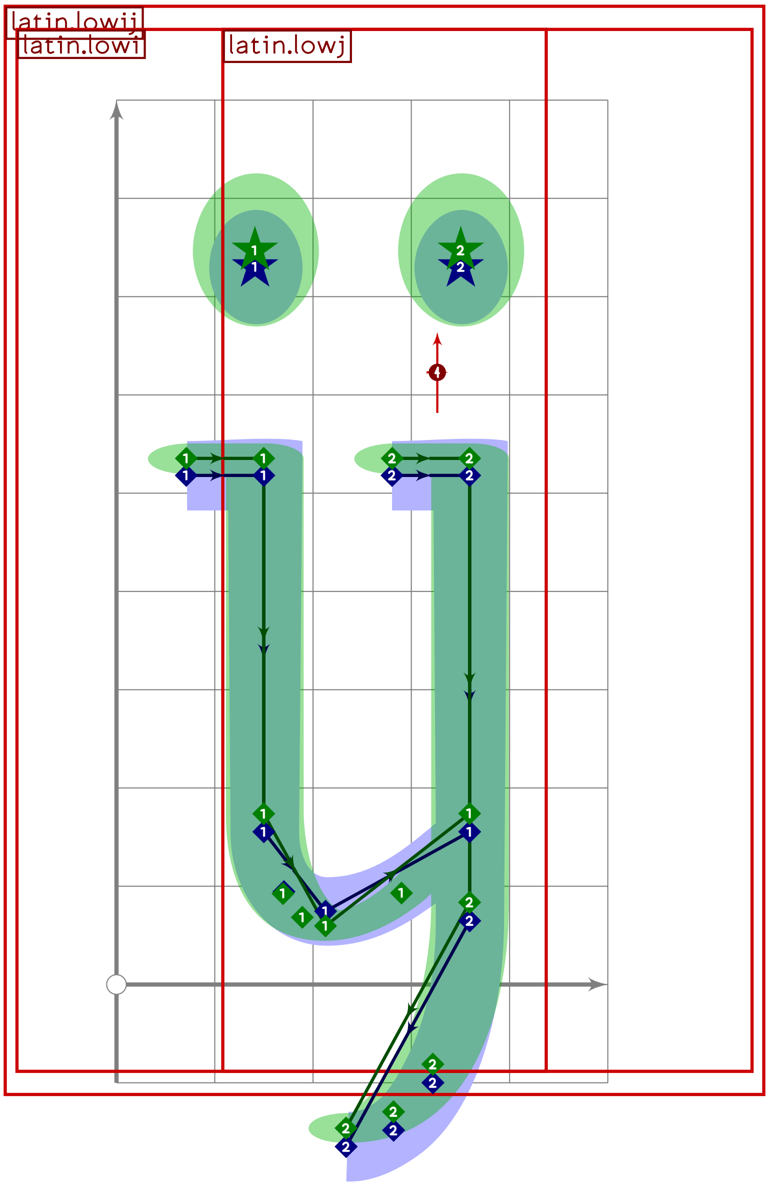


```

vardef latin.lowi =
  push_pbox_toexpand("latin.lowi");
  x2=x3;
  0.85[x1,x2]=450;
  (x2-x1)=0.3(y2-y3);
  x4=x2-15;

  y1=y2=latin_wide_xheight_h;
  y3=latin_wide_low_v;
  y4=0.5[y2,latin_wide_high_v]+mbrush_width;

  push_stroke(z1--z2--z3,(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,1,1);
  set_boserif(0,2,3);

  push_lcblob(fullcircle rotated 45 scaled (mbrush_width*2.7+15)
    shifted (z4 transformed tsu_rescale_xform)
    transformed inverse tsu_rescale_xform);

  push_anchor(anc_wide,
    identity xscaled 0.7 transformed accent_default[anc_wide]);
  tsu_accent.shift_anchors(true)((x4-500,0));
  tsu_accent.shift_anchors(ai=anc_acute)((-55,0));
  tsu_accent.shift_anchors(ai=anc_wide)((-40,0));
  tsu_accent.shift_anchors(ai=anc_tilde)((10,0));
  tsu_accent.shift_anchors(ai=anc_ring)((-10,55));
  expand_pbox;
enddef;
```

U+FF4A
tsuku.uniFF4A



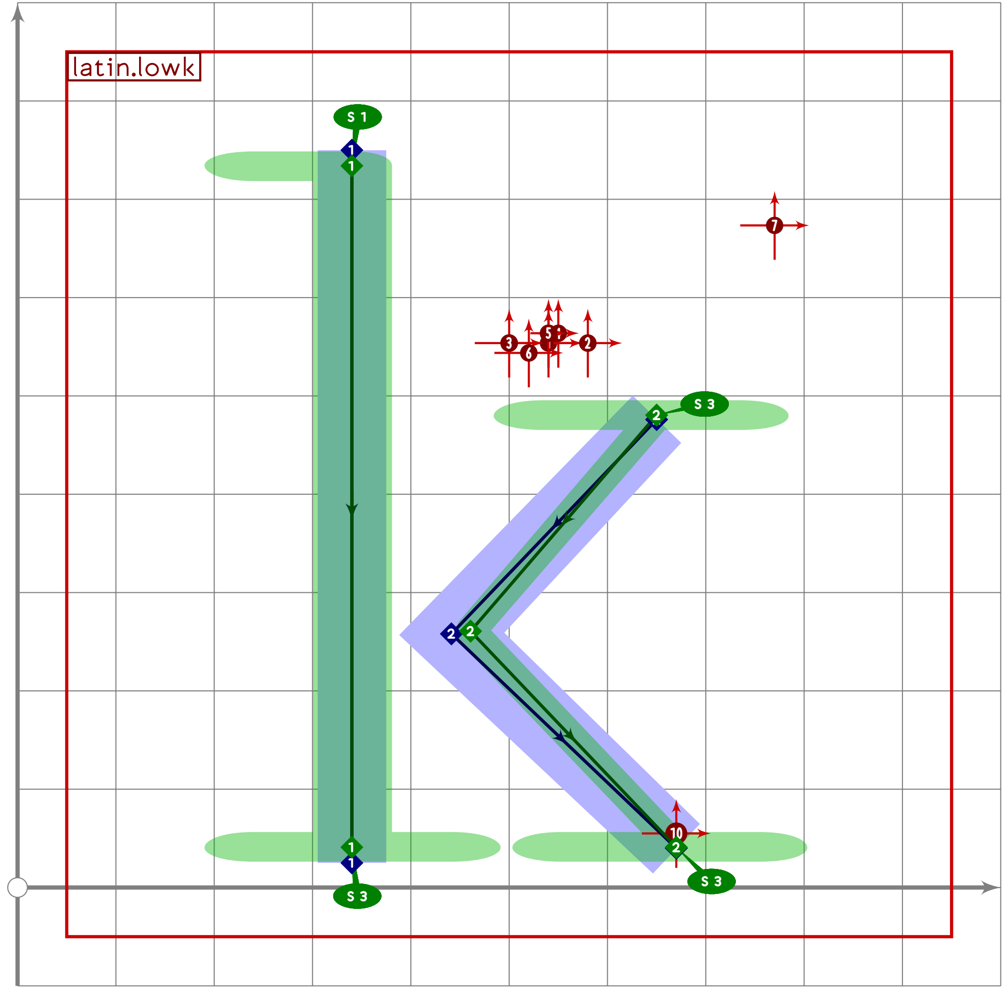
```
vardef latin.lowj =
  push_pbox_toexpand("latin.lowj");
  x2=x3;
  0.85[x1,x2]=450;
  (x2-x1)=0.3(y2-y3);
  x5=x2-15;

  y1=y2=latin_wide_xheight_h;
  y3=latin_wide_low_v;
  y5=0.5[y2,latin_wide_high_v]+mbrush_width;

  z4=z3+(-210,-140);

  push_stroke((z1--z2--(z3+(0,55)))..{curl 0.8}z4,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,1,1);

  push_lcblob(fullcircle scaled (mbrush_width*2.7+15)
    shifted (z5 transformed tsu_rescale_xform)
    transformed inverse tsu_rescale_xform);

  push_anchor(anc_wide,
    identity xscaled 0.7 transformed accent_default[anc_wide]);
  tsu_accent.shift_anchors(true)((x5-500,0));
  tsu_accent.shift_anchors(ai=anc_acute)((-55,0));
  tsu_accent.shift_anchors(ai=anc_wide)((-40,0));
  tsu_accent.shift_anchors(ai=anc_tilde)((10,0));
  tsu_accent.shift_anchors(ai=anc_ring)((-10,55));
  tsu_accent.shift_anchors(ai=anc_lower)((80,-10));
  tsu_accent.shift_anchors(ai=anc_lower_connect)((-40,-80));
  expand_pbox;
enddef;
```



```

vardef latin.lowij =
```

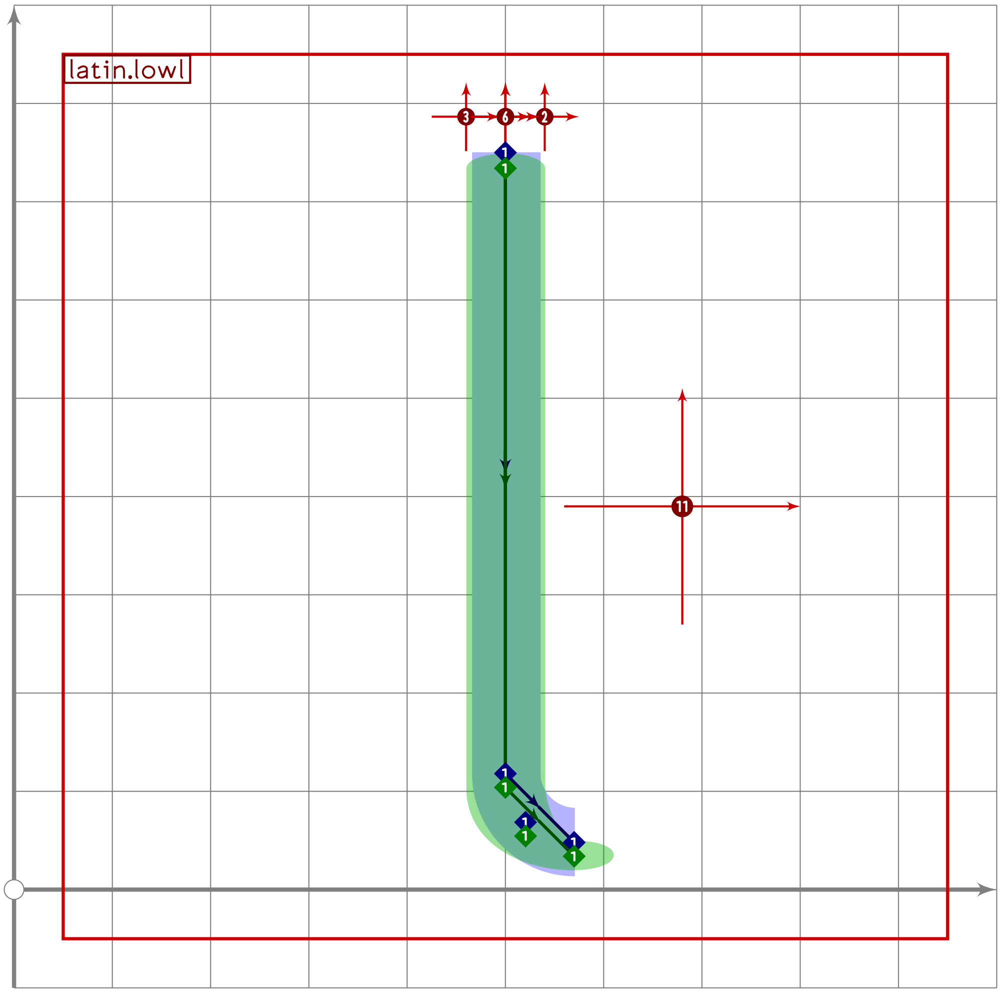
U+FF4B
tsuku.uniFF4B

```
  push_pbox_toexpand("latin.lowij");
  tsu_xform(identity shifted (-200,0))(latin.lowi);
  numeric x[],y[];
  tsu_xform(identity shifted (150,0))(latin.lowj);
  replace_lcblob(-1)(get_lcblob(0) shifted (-350,0));
  numeric x[],y[];
  z1=point 1.7 of get_strokep(-1);
  x2=0.3[x1,x3];
  y2=vmetric(0.05);
  z3=point 1.8 of get_strokep(0);
  replace_strokep(-1)((subpath (0,1) of oldp)--
    z1{dir 273}..z2{right}..{curl 0.3}z3);
  replace_strokeq(-1)((subpath (0,1) of oldq)--(1.6,1.6)--(1.6,1.6)--(1,1));
  set_boserif(-1,2,whatever);
  expand_pbox;
enddef;
```

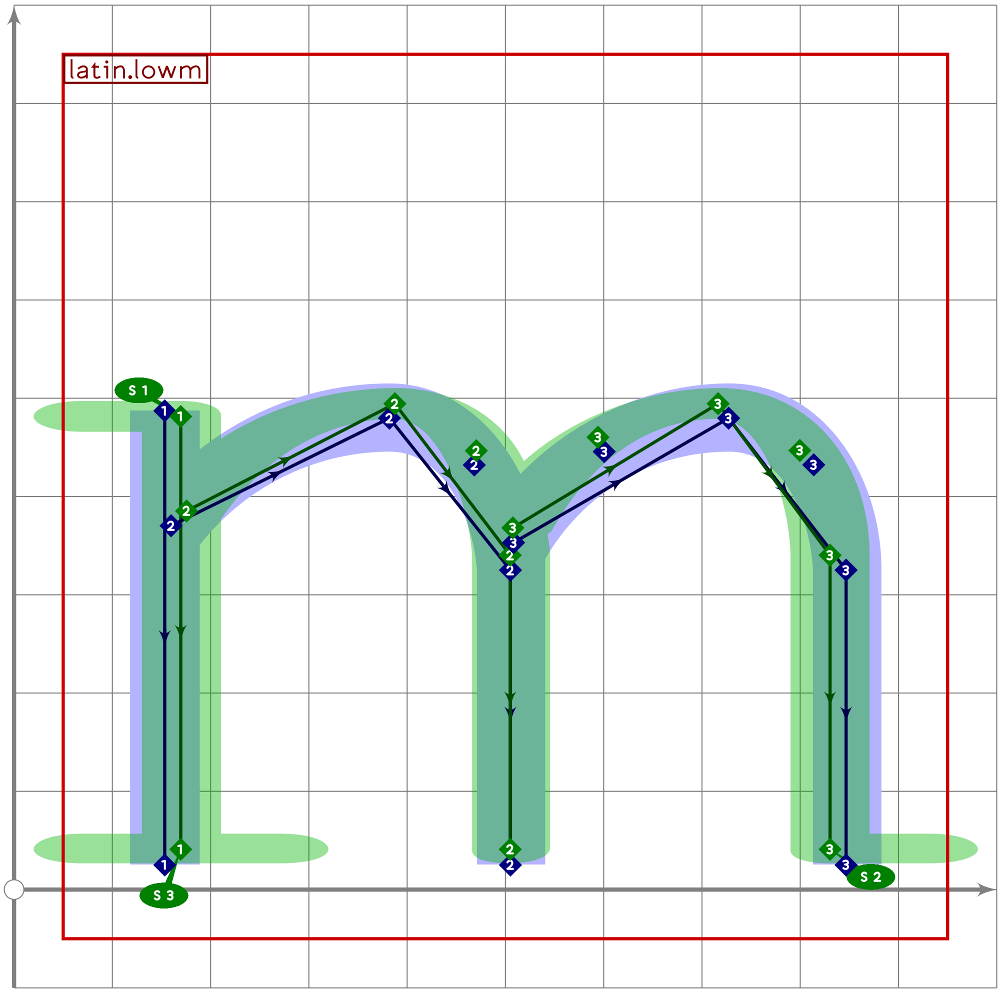


```

vardef latin.lowk =
  push_pbox_toexpand("latin.lowk");
  z1=(340,latin_wide_high_v);
  z2=(340,latin_wide_low_v);
  z3=(650,(latin_wide_xheight_v+latin_wide_xheight_h)/2);
  x4=340+mbrush_width*if sharp_corners: 2.7 else: 2.3 fi;
  y4=(y3+y5)/2;
  z5=(670,0.5[latin_wide_low_h,latin_wide_low_v]);

  push_stroke(z1--z2,(1.6,1.6)--(1.6,1.6));
  if not do_italic_hook:
    set_boserif(0,0,1);
    set_boserif(0,1,3);
  fi;

  push_stroke(z3--z4--z5,(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,1,1);
  set_boserif(0,0,if do_italic_hook: 1 else: 3 fi);
  if not do_italic_hook: set_boserif(0,2,3); fi;
  set_boalternate(0);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))((40,0));
  push_anchor(anc_wide,identity xscaled 0.8
    transformed accent_default[anc_wide] shifted (50,10));
  push_anchor(anc_tilde,identity xscaled 0.7
    transformed accent_default[anc_tilde] shifted (40,10));
  push_anchor(anc_ring,accent_default[anc_ring] shifted (20,-10));
  push_anchor(anc_lower_connect,
    accent_default[anc_lower_connect] shifted (170,0));
  expand_pbox;
enddef;
```

U+FF4C
tsuku.uniFF4C



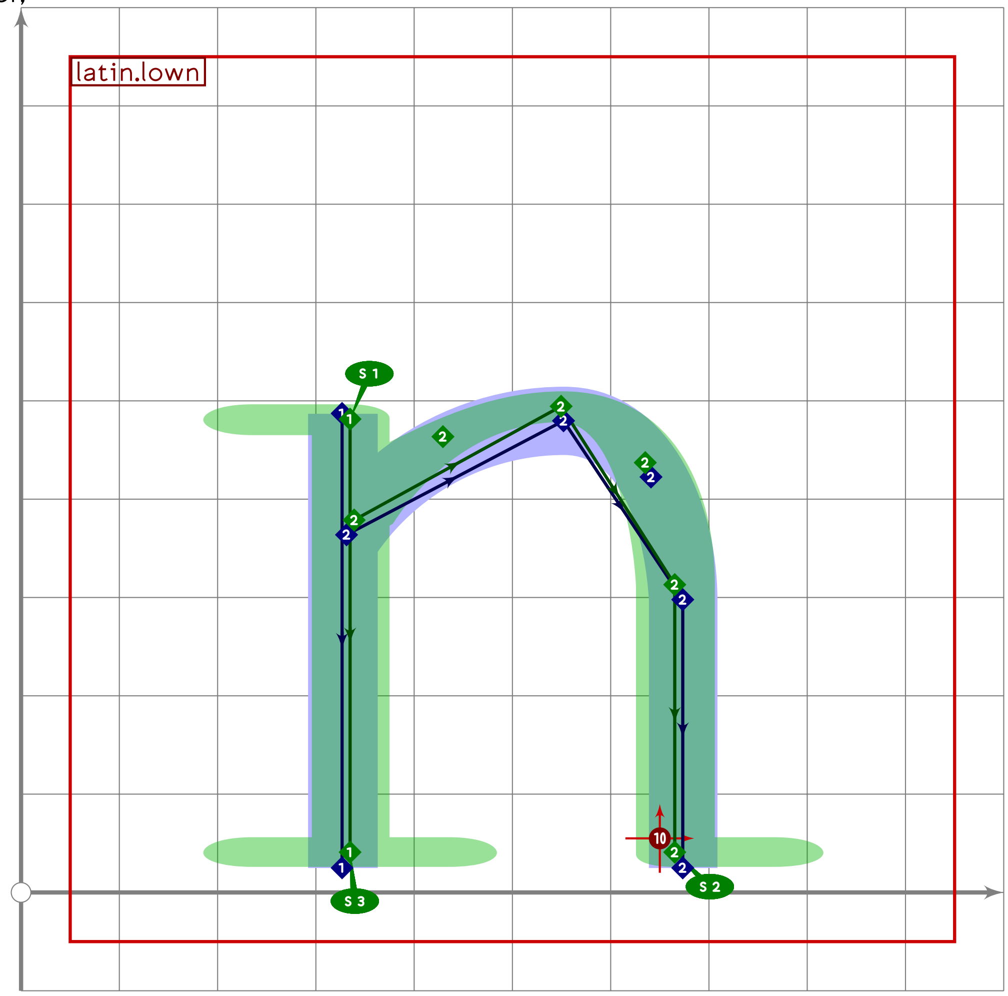
```
vardef latin.lowl =
  push_pbox_toexpand("latin.lowl");
  x1=x2=500;
  x3=570;

  y1=latin_wide_high_v;
  y2=y3+(x3-x2);
  y3=latin_wide_low_r;

  push_stroke(z1--z2{down}..{right}z3,(1.6,1.6)--(1.6,1.6)--(1.6,1.6));

  tsu_accent.shift_anchors((ypart olda>vmetric(0.52))
                         and not (ai=anc_caron_comma))
    (((0,0) transformed tsu_xf.cap_upper_accent)-
     ((0,0) transformed accent_default[anc_upper]));
  tsu_accent.shift_anchors(ai=anc_centre)((180,0));
  expand_pbox;
enddef;
```



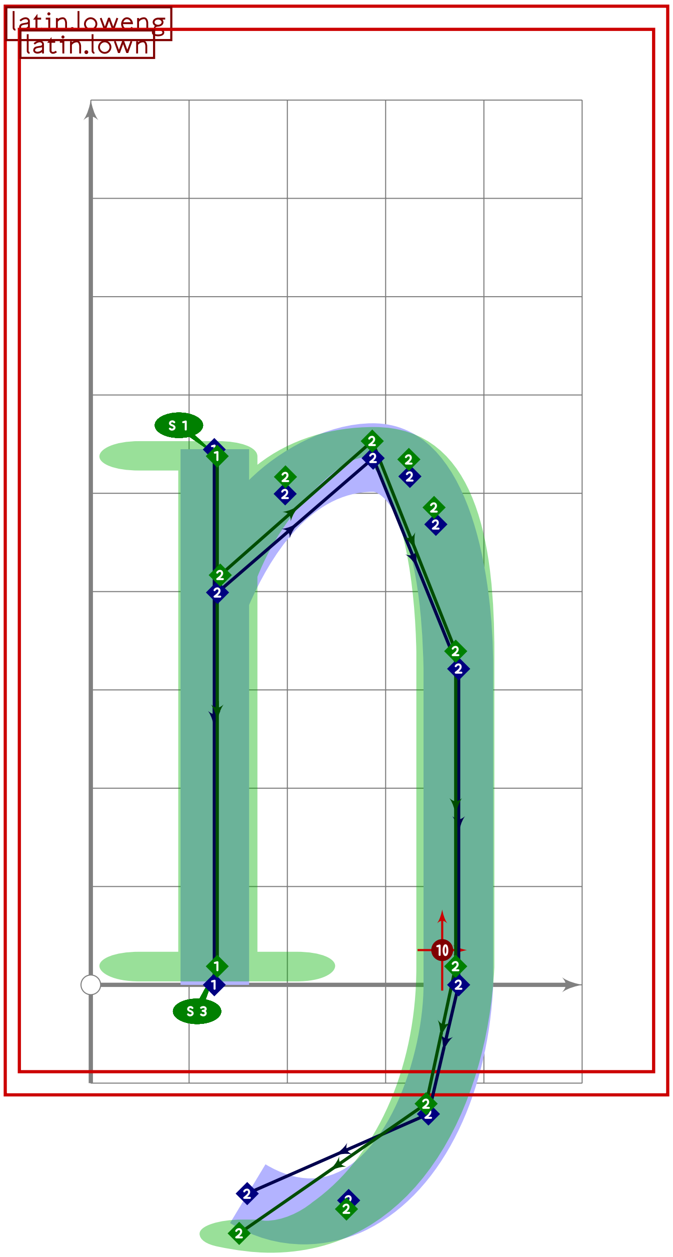
```
vardef latin.lowm =
  push_pbox_toexpand("latin.lowm");
  (x1+x9)/2=500;
  (x9-x1)*2=(y1-y2)*3;
  (x5-x1)=(x9-x5)*1.03;
  x2=x1=x3;
  x4=0.65[x3,x5];
  x6=x5;
  x8=0.65[x6,x9];
  x9=x10;

  y1=latin_wide_xheight_v;
  y2=y6=y10=latin_wide_low_v;
  y3=0.74[y2,y4];
  y4=y8=latin_wide_xheight_r;
  y5=y9=0.66[y2,y4];

  push_stroke(z1--z2,(1.6,1.6)--(1.6,1.6));
  set_boserif(0,0,if do_italic_hook: 11 else: 1 fi);
  if not do_italic_hook: set_boserif(0,1,3); fi;
```

U+FF4E
tsuku.uniFF4E

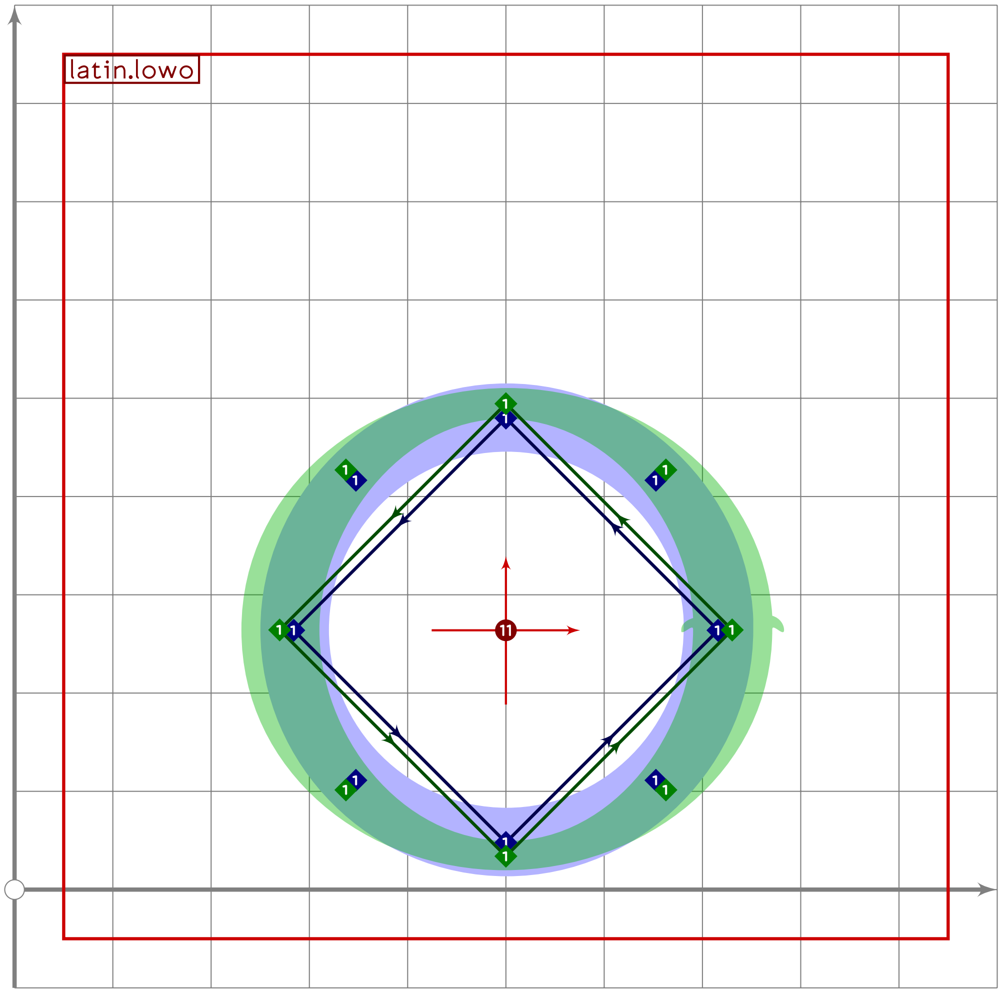
```

  push_stroke(z3..z4{right}..z5{dir 275}--z6,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(subpath (0.04,3) of oldp);

  z7=get_strokep(0) intersectionpoint (z3--z9);

  push_stroke(z7..z8{right}..z9{dir 271}--z10,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(subpath (0.04,3) of oldp);
  set_boserif(0,3,if do_italic_hook: 11 else: 2 fi);
  expand_pbox;
enddef;
```



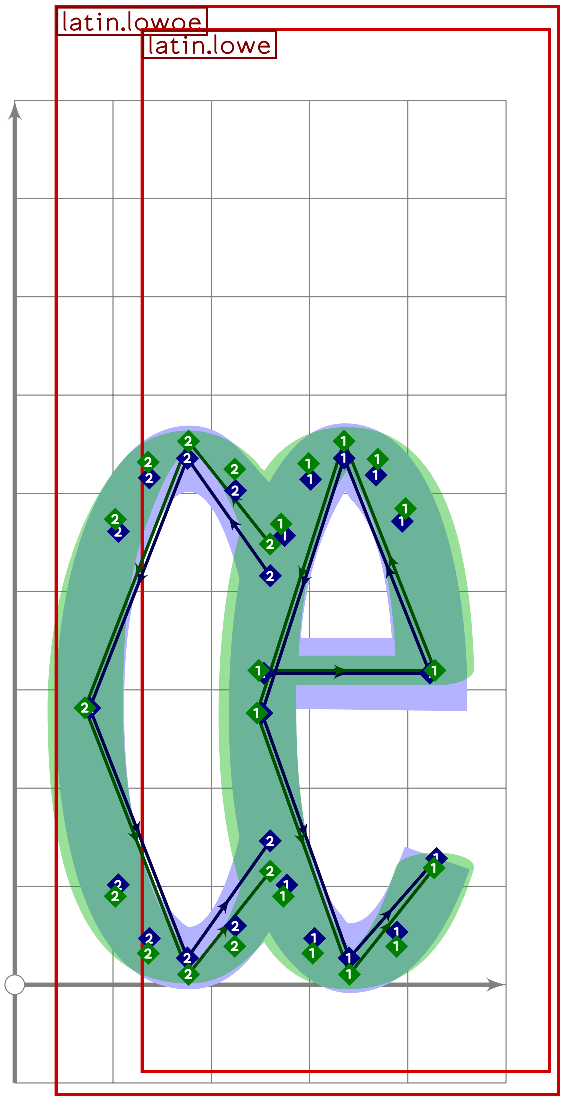
```

vardef latin.lown =
  push_pbox_toexpand(”latin.lown”);
  (x1+x5)/2=500;
  (x5-x1)=(y1-y2)*0.75;
  x2=x1=x3;
  x4=0.65[x3,x5];
  x6=x5;

  y1=latin_wide_xheight_v;
  y2=y6=latin_wide_low_v;
  y3=0.73[y2,y4];
  y4=latin_wide_xheight_r;
  y5=0.60[y2,y4];

  push_stroke(z1--z2,(1.6,1.6)--(1.6,1.6));
  set_boserif(0,0,if do_italic_hook: 11 else: 1 fi);
  if not do_italic_hook: set_boserif(0,1,3); fi;

  push_stroke(z3..z4{right}..z5{dir 273}--z6,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(subpath (0.03,3) of oldp);
  set_boserif(0,3,if do_italic_hook: 11 else: 2 fi);

  push_anchor(anc_lower_connect,
    accent_default[anc_lower_connect] shifted (150,0));
  expand_pbox;
enddef;
```



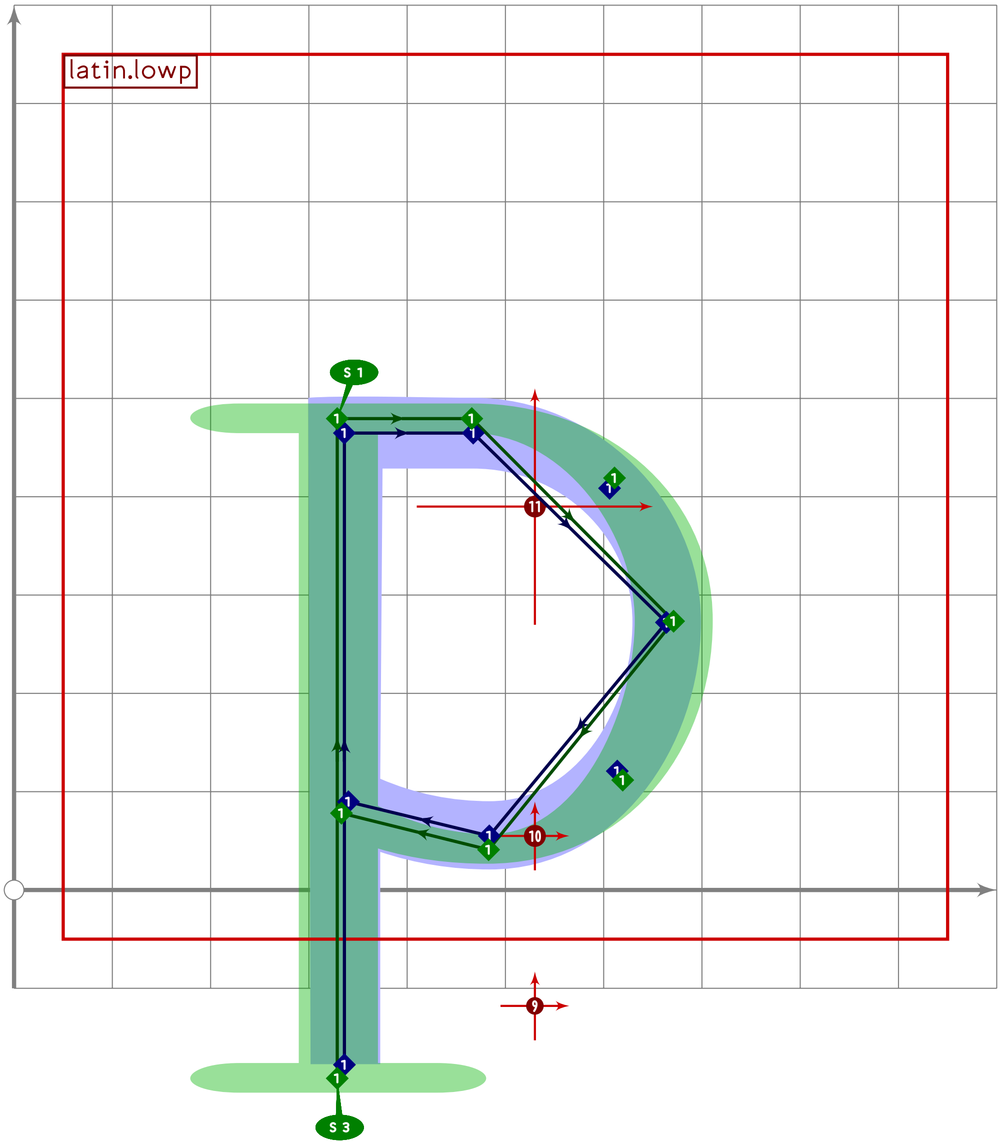
```
vardef latin.loweng =
  push_pbox_toexpand(”latin.loweng”);
  latin.lown;
  x7=x6-300;
  y7=latin_wide_desc_h;
  replace_strokep(0)(oldp{dir 266}..{curl 0.8}z7);
  replace_strokep(0)(insert_nodes(oldp)(3.3));
  replace_strokeq(0)(oldq–(1.6,1.6)–(1.6,1.6));
  set_boserif(0,3,whatever);
  expand_pbox;
enddef;
```



```

vardef latin.lowo =
  push_pbox_toexpand(”latin.lowo”);
  push_anchor(anc_centre,identity
    scaled ((latin_wide_xheight_r-latin_wide_low_r)/200)
    shifted (xpart centre_pt,(latin_wide_xheight_r+latin_wide_low_r)/2));
  push_stroke(((1,0)..(0,1)..(-1,0)..(0,-1)..cycle)
    scaled ((latin_wide_xheight_r-latin_wide_low_r)/2)
    shifted (xpart centre_pt,(latin_wide_xheight_r+latin_wide_low_r)/2),
    (1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)–cycle);
  expand_pbox;
enddef;
```



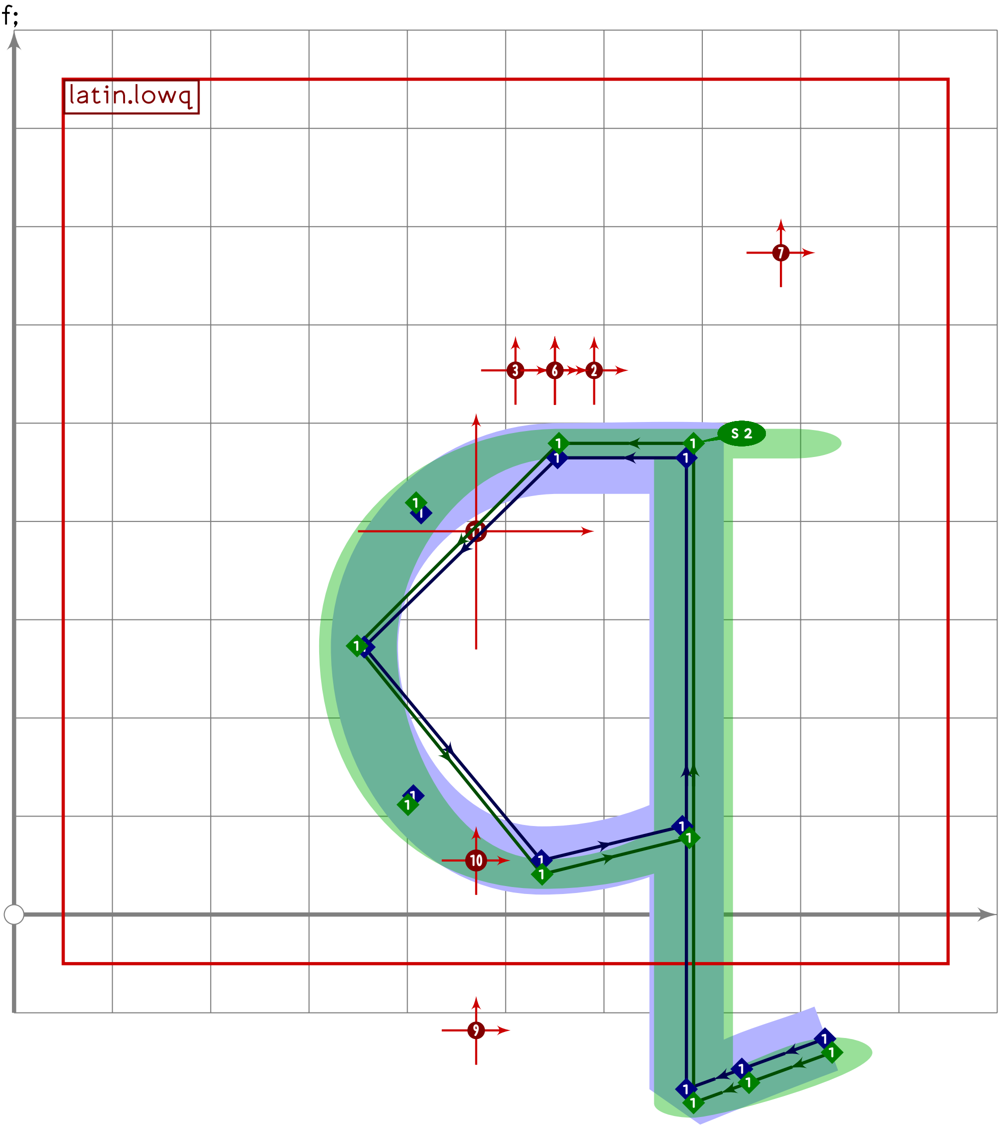
```

vardef latin.lowoe =
  push_pbox_toexpand("latin.lowoe");
  tsu_xform(identity shifted (190,0))(latin.lowe);
  push_stroke(((1,0)..(0,1)..(-1,0)..(0,-1)..(1,0))
    scaled ((latin_wide_xheight_r-latin_wide_low_r)/2)
    shifted (340,0.5[latin_wide_xheight_r,latin_wide_low_r]),
    (1.2,1.2)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.2,1.2));
  replace_strokep(0)(subpath (0.03+xpart (oldp intersectiontimes
                                 (subpath (2,infinity) of get_strokep(-1))),
                             3.97-xpart ((reverse oldp) intersectiontimes
                                      reverse get_strokep(-1)))
               of oldp);
  expand_pbox;
enddef;
```



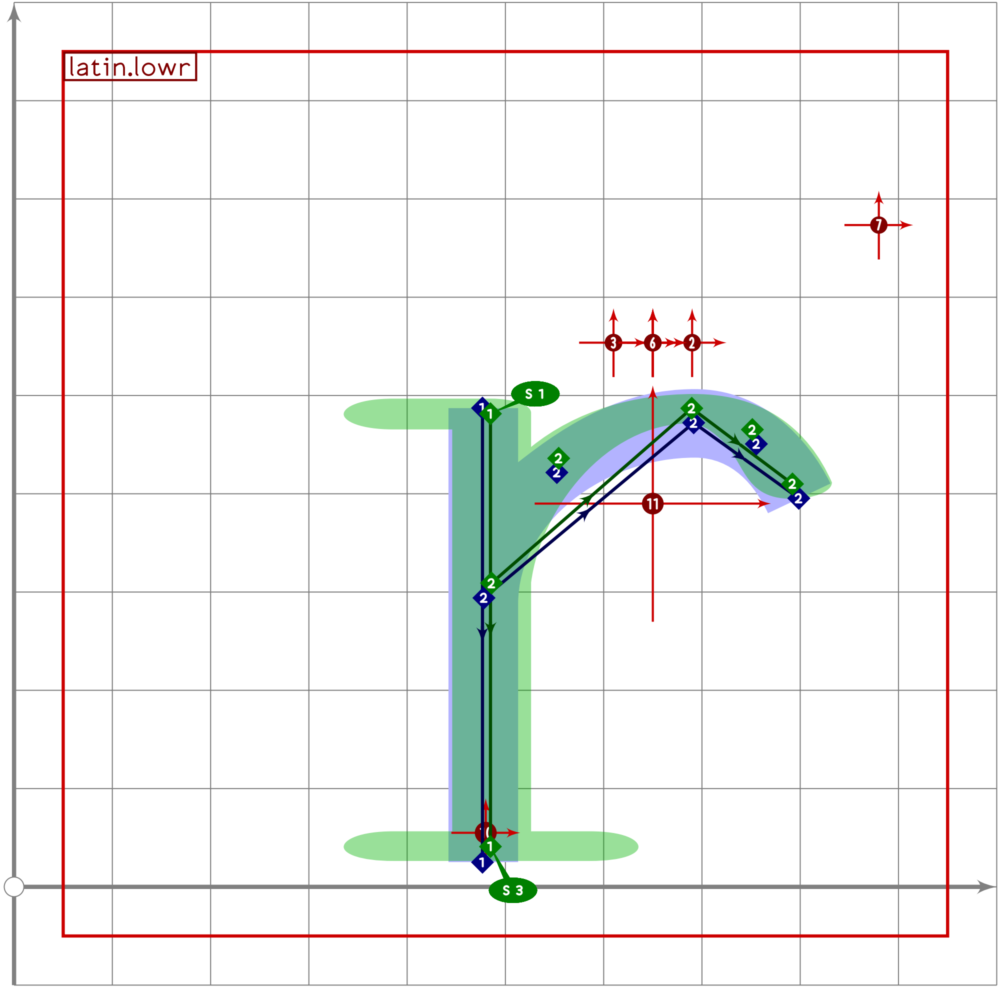
```
vardef latin.lowp =
  push_pbox_toexpand(”latin.lowp”);
  (x1+x4)/2=500;
  (x4-x1)=(y2-y1)*0.51;
  x2=x1=x6;
  x3=0.4[x2,x4];
  x5=0.45[x2,x4];

  y1=latin_wide_desc_v;
  y2=y3=latin_wide_xheight_h;
  y4=0.47[y3,y5];
  y5=latin_wide_low_h;
  y6=0.91[y3,y5];

  push_stroke(z1--z2{right}..{right}z3..{down}z4..{left}z5..z6,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--
      (1.6,1.6)--(1.6,1.6));
```

U+FF51
tsuku.uniFF51

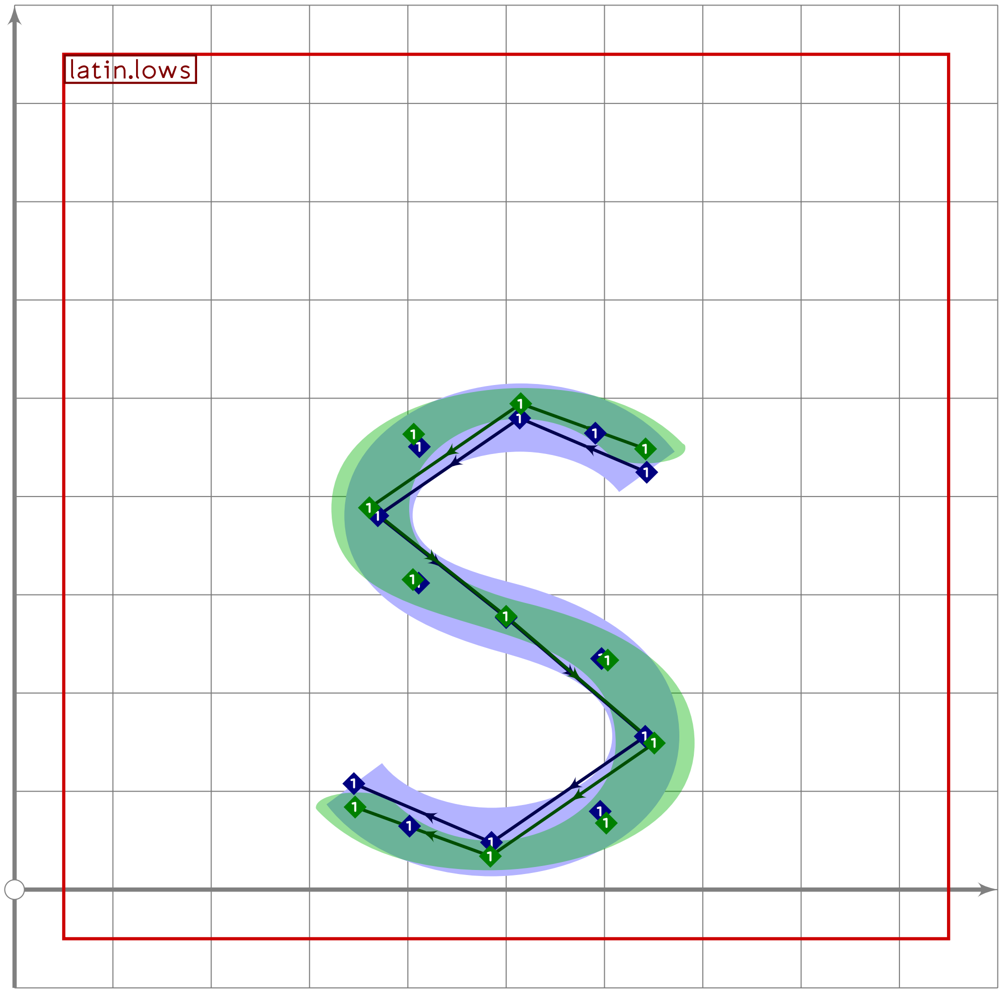
```
  replace_strokep(0)(subpath (0,4.97) of oldp);
  set_botip(0,1,1);
  if not do_italic_hook: set_boserif(0,0,3); fi;
  set_boserif(0,1,1);

  tsu_accent.shift_anchors(ypart olda<vmetric(0.52))((30,0));
  expand_pbox;
enddef;
```



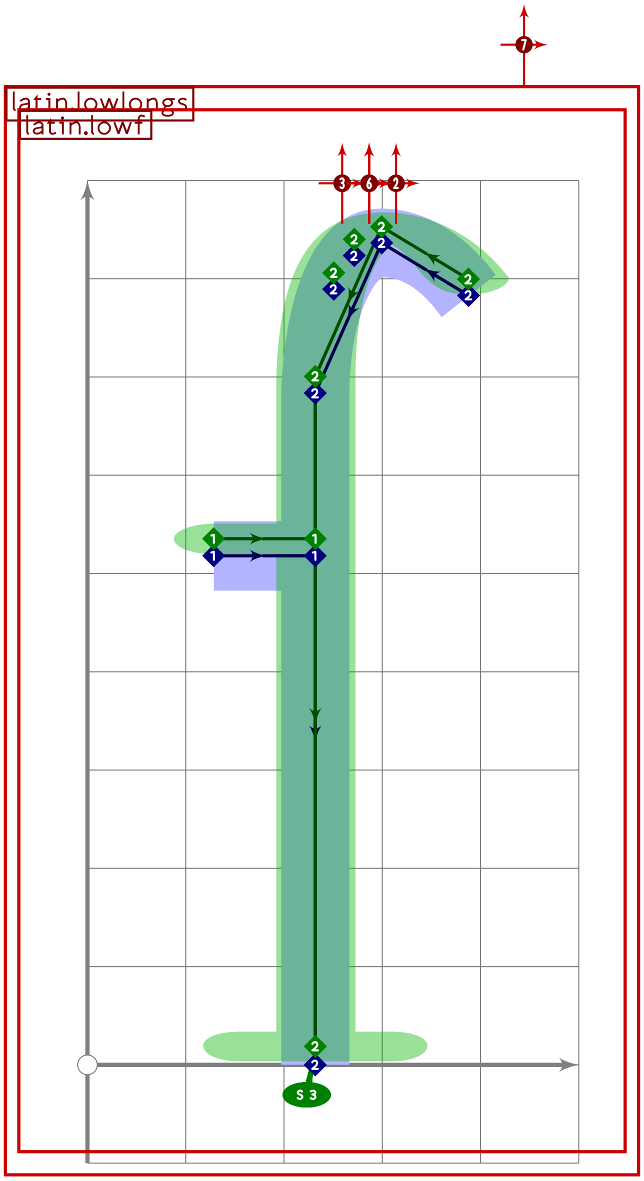
```

vardef latin.lowq =
  push_pbox_toexpand("latin.lowq");
  (x1+x4)/2=520;
  (x1-x4)=(y2-y1)*0.51;
  x2=x1=x6;
  x3=0.4[x2,x4];
  x5=0.45[x2,x4];

  y1=latin_wide_desc_v;
  y2=y3=latin_wide_xheight_h;
  y4=0.47[y3,y5];
  y5=latin_wide_low_h;
  y6=0.91[y3,y5];

  z0=z1+150*(dir 20);

  push_stroke(z0--(0.6[z0,z1])--z1--z2{left}..
    {left}z3..{down}z4..{right}z5..z6,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--
      (1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(subpath (0,6.97) of oldp);
  set_botip(0,2,0);
  set_botip(0,3,1);
  if not do_italic_hook: set_boserif(0,3,2); fi;

  tsu_accent.shift_anchors(ypart olda<vmetric(0.52))((-30,0));
  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))((50,0));
  expand_pbox;
enddef;
```



```
vardef latin.lowr =
  push_pbox_toexpand("latin.lowr");
  (x1+x5)/2=650;
  (x5-x1)=(y1-y2)*0.75;
  x2=x1=x3;
  x4=0.62[x3,x5];

  y1=latin_wide_xheight_v;
  y2=latin_wide_low_v;
  y3=0.58[y2,y4];
  y4=0.5[latin_wide_xheight_h,latin_wide_xheight_r];
  y5=0.60[y2,y4];

  push_stroke(z1--z2,(1.6,1.6)--(1.6,1.6));
  set_boserif(0,0,if do_italic_hook: 11 else: 1 fi);
  if not do_italic_hook: set_boserif(0,1,3); fi;

  push_stroke(z3..z4{right}..{dir 273}z5,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(subpath (0.03,1.6) of oldp);

  tsu_accent.shift_anchors(ypart olda>vmetric(0.05))((0.5[x3,x5]-500,0));
  tsu_accent.shift_anchors(ai=anc_lower_connect)((-20,0));
  expand_pbox;
enddef;
```

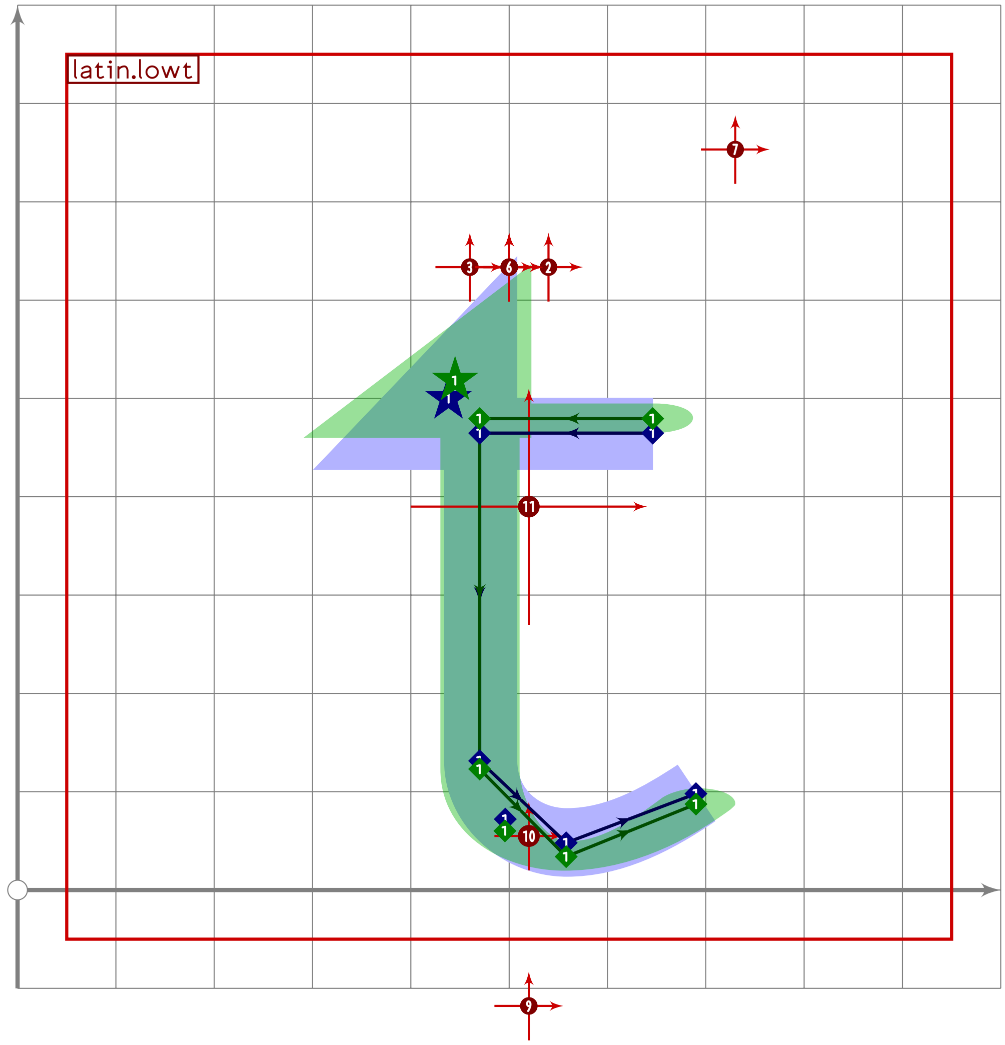


```

vardef latin.lows =
  push_pbox_toexpand("latin.lows");
  transform ta,tb;
  path mycurve;

  mycurve:=(1,0)..(0,1)..(-1,0);

  y2=latin_wide_xheight_r;
  y0=y3=0.77[y6,y2];
  y4=0.53[y6,y2];
  y5=y8=0.25[y6,y2];
  y6=latin_wide_low_r;

  0.48[x1,x7]=0.48[x2,x6]=0.48[x3,x5]=x4=500;
  x5-x1=5;
  x5-x7=(y2-y6)*0.67;

  (point 0 of mycurve) transformed ta=z0;
  (point 0.35 of mycurve) transformed ta=z1;
  (point 1 of mycurve) transformed ta=z2;
  (point 2 of mycurve) transformed ta=z3;
  xypart ta=0;

  (point 0 of mycurve) transformed tb=z8;
  (point 0.35 of mycurve) transformed tb=z7;
  (point 1 of mycurve) transformed tb=z6;
  (point 2 of mycurve) transformed tb=z5;

  if sharp_corners:
    mycurve:=subpath (0.29,2) of mycurve;
  else:
    mycurve:=subpath (0.38,2) of mycurve;
  fi;

  push_stroke((mycurve transformed ta)..z4..(reverse mycurve transformed tb),
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--
      (1.6,1.6)--(1.6,1.6));
  expand_pbox;
enddef;
```



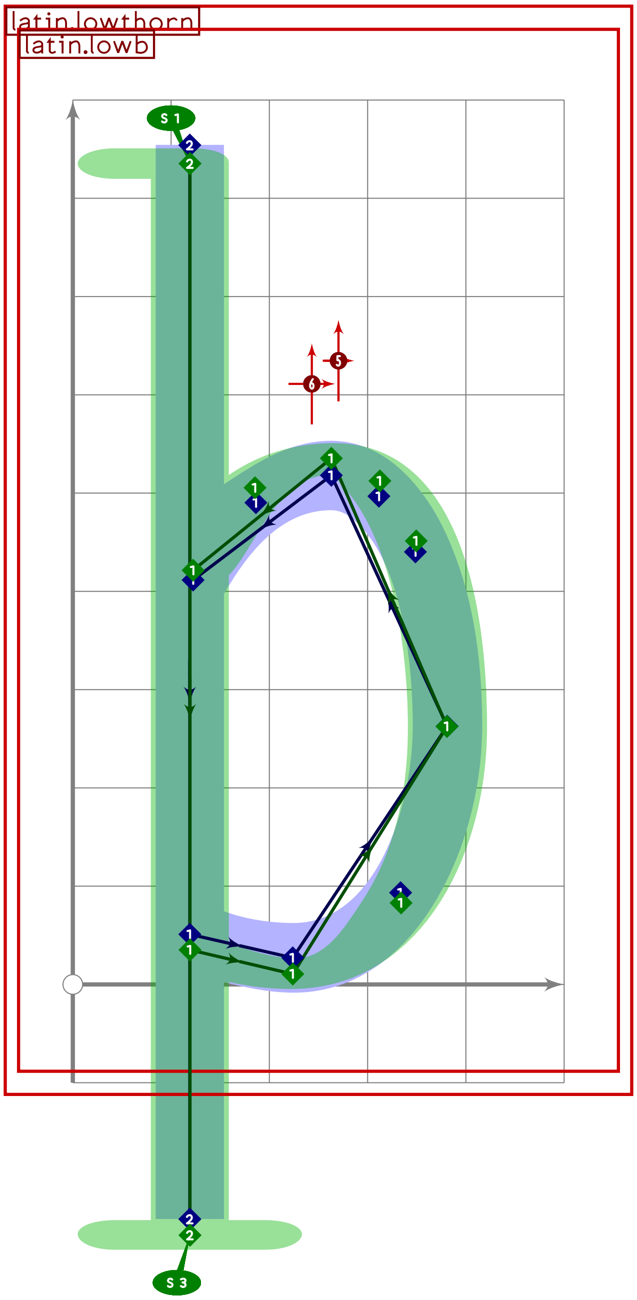
```
vardef latin.lowlongs =
  push_pbox_toexpand("latin.lowlongs");
  latin.lowf;
  replace_strokep(-1)(subpath (0,0.52) of oldp);
  expand_pbox;
enddef;
```

U+FF54
tsuku.uniFF54



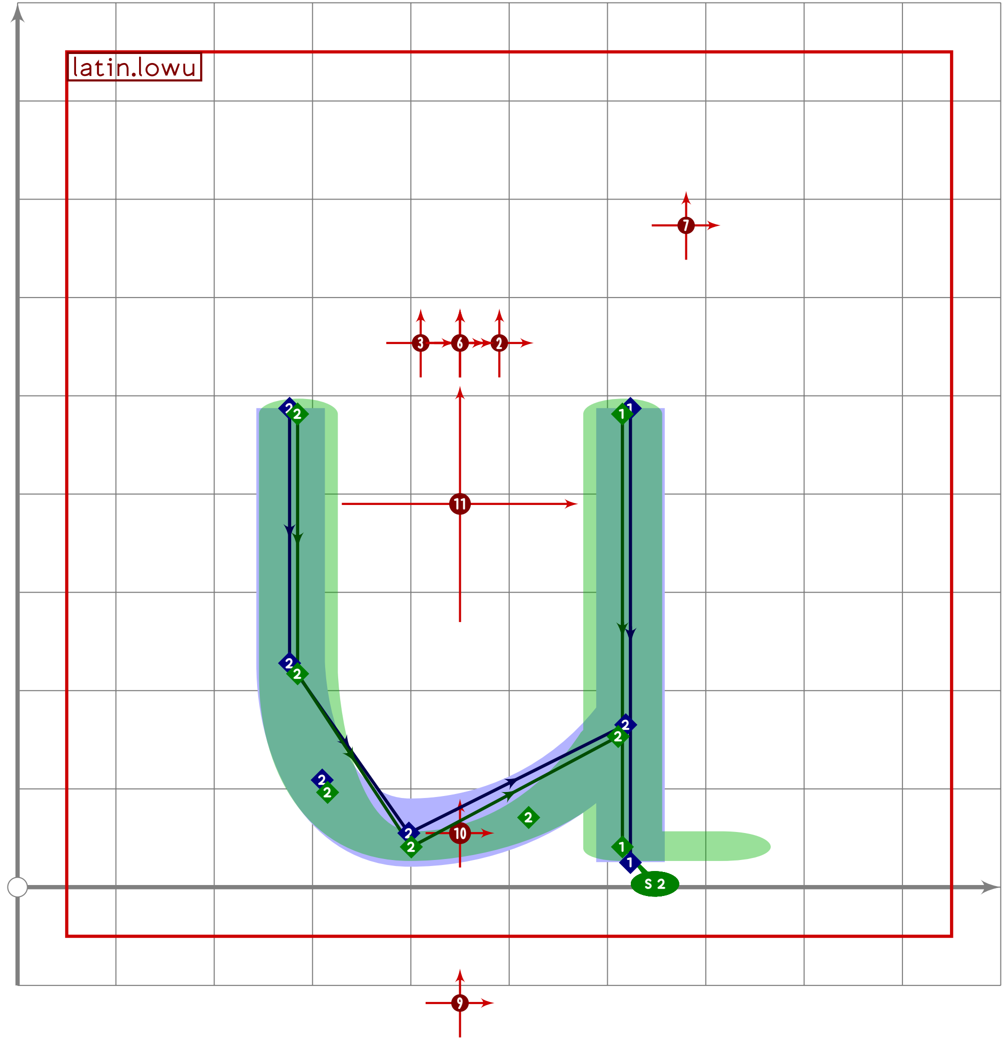
```

vardef latin.lowt =
  push_pbox_toexpand("latin.lowt");
  x2=x3=x10=470;
  x1=0.2[x5,x2];
  x4=0.6[x5,x2];
  x5-x2=220;

  y1=y2=y6=latin_wide_xheight_h;
  y3=0.2[y4,y1];
  y4=latin_wide_low_r;
  y5=0.12[y4,y1];
  y10=vmetric(0.83);

  if is_proportional:
    z10-z6=whatever*dir 58;
  else:
    z10-z6=whatever*dir 47;
  fi;

  if tsu_pbrush_size<30:
    push_stroke(z1--z6--z10--z3{down}..z4{right}..{curl 0.2}z5,
      (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
    set_botip(0,1,1);
    set_botip(0,2,1);
  else:
    push_stroke(z1--z2--z3{down}..z4{right}..{curl 0.2}z5,
      (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
    set_botip(0,1,0);
  fi;

  push_stroke(z6--z1,(0,0)--(0,0));

  x7=x8=x2;
  x9=x6;

  y7=y9=y2;

  if is_proportional:
    z8-z9=whatever*dir 58;
  else:
    z8-z9=whatever*dir 47;
  fi;

  if tsu_pbrush_size>=30:
    begingroup
      save t; transform t;
      t:=tsu_rescale_xform;
      push_lcblob(((z7 transformed t)+(mbrush_width,-mbrush_height))--
                  ((z8 transformed t)+(mbrush_width,mbrush_height))--
                  ((z9 transformed t)+(-mbrush_width,-mbrush_height))--cycle);
      replace_lcblob(0)(oldblob transformed inverse t);
    endgroup;
  fi;

  tsu_accent.shift_anchors(ypart olda>vmetric(0.52))
    ((0,vmetric(0.12)-vmetric(0)));
  tsu_accent.shift_anchors(ypart olda<vmetric(0.52))
    ((20,0));
  expand_pbox;
enddef;
```

U+00FE
tsuku.thorn



```
vardef latin.lowthorn =
  push_pbox_toexpand("latin.lowthorn");
  latin.lowb;
  set_botip(0,1,whatever);
  set_boserif(0,0,whatever);
  push_stroke((point 0 of get_strokep(0))--
    (xpart point 0 of get_strokep(0),latin_wide_desc_v),
    (1.6,1.6)--(1.6,1.6));
  set_boserif(0,0,1);
  set_boserif(0,1,3);
  replace_strokep(-1)(subpath (1,infinity) of oldp);
  replace_strokeq(-1)(subpath (1,infinity) of oldq);
  expand_pbox;
```

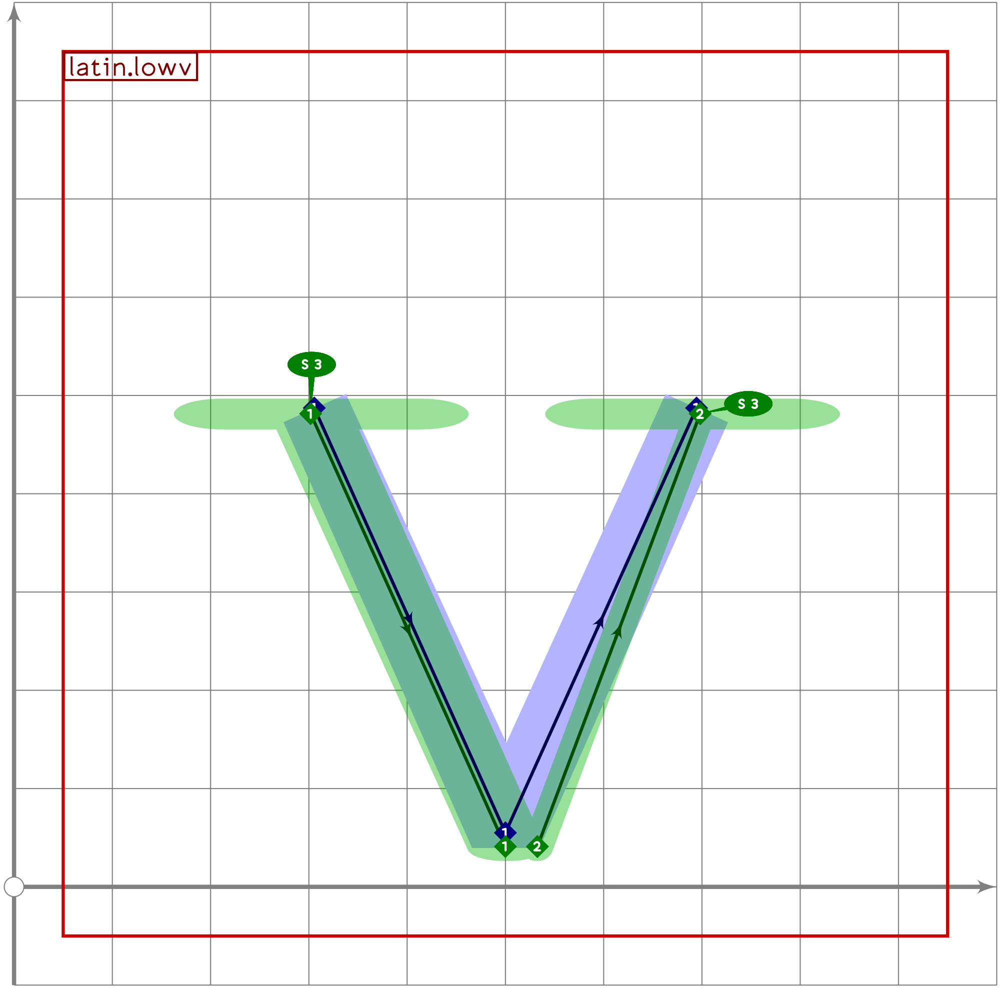
```
enddef;
```



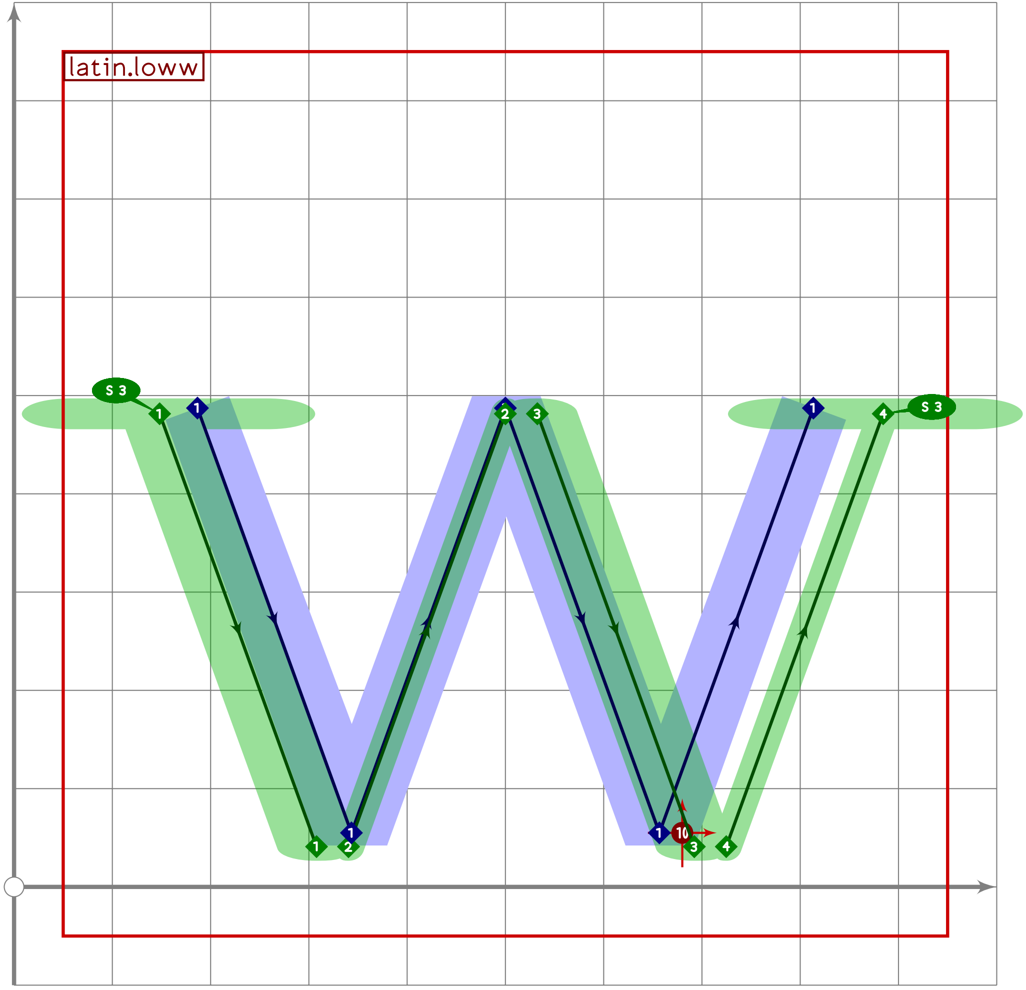
```

vardef latin.lowu =
  push_pbox_toexpand(”latin.lowu”);
  (x1+x5)/2=450;
  (x1-x5)=(y2-y1)*0.75;
  x2=x1=x3;
  x4=0.65[x3,x5];
  x6=x5;

  y1=latin_wide_low_v;
  y2=y6=latin_wide_xheight_v;
  y3=0.73[y2,y4];
  y4=latin_wide_low_h;
  y5=0.60[y2,y4];

  push_stroke(z2--z1,(1.6,1.6)--(1.6,1.6));
  set_boserif(0,1,if do_italic_hook: 11 else: 2 fi);

  push_stroke(reverse(z3..z4{left}..z5{dir 93}--z6),
```

U+FF56
tsuku.uniFF56

```
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(subpath (0,2.97) of oldp);
  if do_italic_hook: set_boserif(0,0,11); fi;

  tsu_accent.shift_anchors(true)((-50,0));
  expand_pbox;
enddef;
```

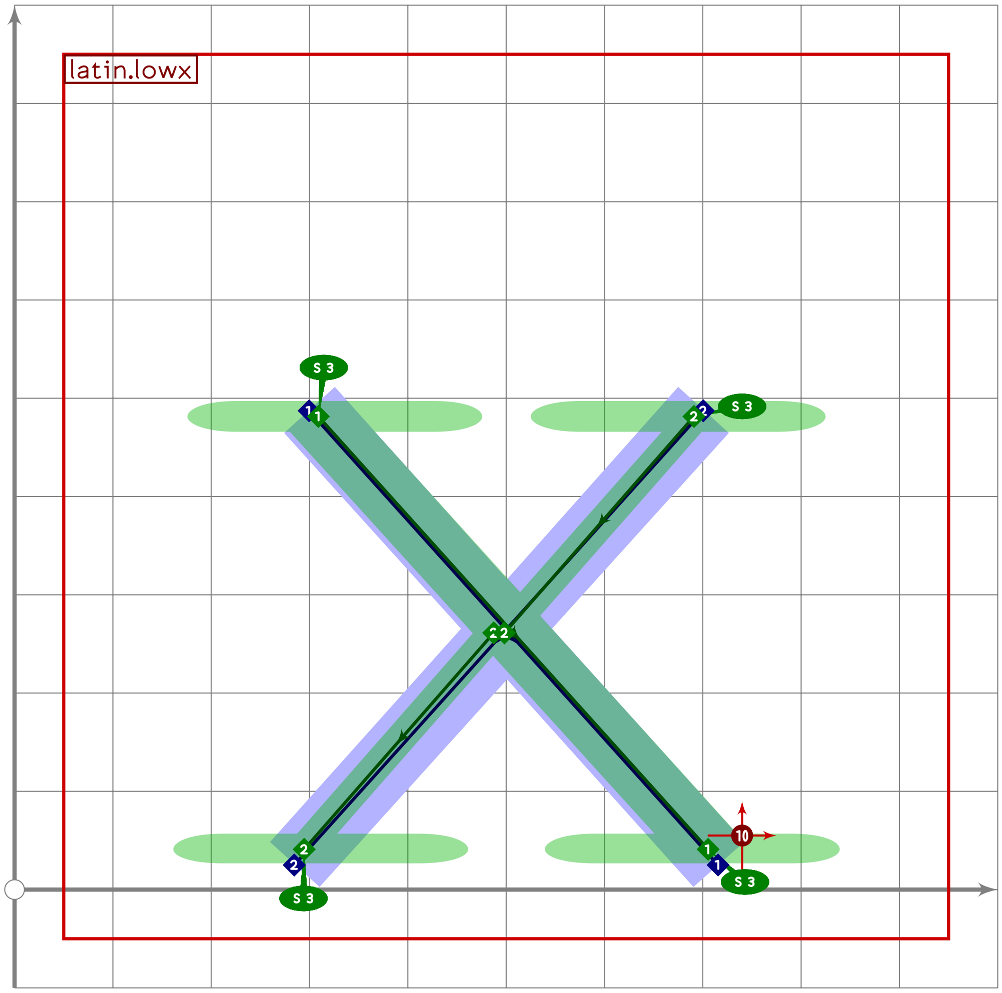


```

vardef latin.lowv =
  push_pbox_toexpand("latin.lowv");
  (x1+x3)/2=x2=500;

  y1=y3=latin_wide_xheight_v;
  y2=latin_wide_low_h;

  (x3-x1)=(y1-y2)*0.9;

  if do_alternation:
    push_stroke(z1--z2,(1.6,1.6)--(1.6,1.6));
    set_boserif(0,0,3);

    push_stroke((z2+alternate_adjust*right)–z3,(1.6,1.6)–(1.6,1.6));
    set_boserif(0,1,3);
    set_boalternate(0);
  else:
    push_stroke(z1–z2–z3,(1.6,1.6)–(1.6,1.6)–(1.6,1.6));
    set_botip(0,1,0);
    set_boserif(0,0,1);
  fi;
  expand_pbox;
enddef;
```



```

vardef latin.loww =
  push_pbox_toexpand("latin.loww");
  (x1+x5)/2=(x2+x4)/2=x3=500;
  (x3-x2)=(x2-x1);

  y1=y3=y5=latin_wide_xheight_v;
  y2=y4=latin_wide_low_h;

  (x5-x1)=(y1-y2)*1.45;

  if do_alternation:
    push_stroke((z1--z2) shifted (alternate_adjust*left),
      (1.6,1.6)--(1.6,1.6));
    set_boserif(0,0,3);

    push_stroke(z2--z3,
      (1.6,1.6)--(1.6,1.6));
    set_boalternate(0);

    push_stroke((z3--z4) shifted (alternate_adjust*right),
      (1.6,1.6)--(1.6,1.6));

    push_stroke((z4--z5) shifted (alternate_adjust*right*2),
      (1.6,1.6)--(1.6,1.6));
    set_boserif(0,1,3);
    set_boalternate(0);
  else:
    push_stroke(z1--z2--z3--z4--z5,
      (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
    set_botip(0,1,0);
    set_botip(0,2,0);
    set_botip(0,3,0);
    set_boserif(0,0,1);
  fi;

  tsu_accent.shift_anchors(ai=anc_lower_connect)((180,0));
  expand_pbox;
enddef;
```

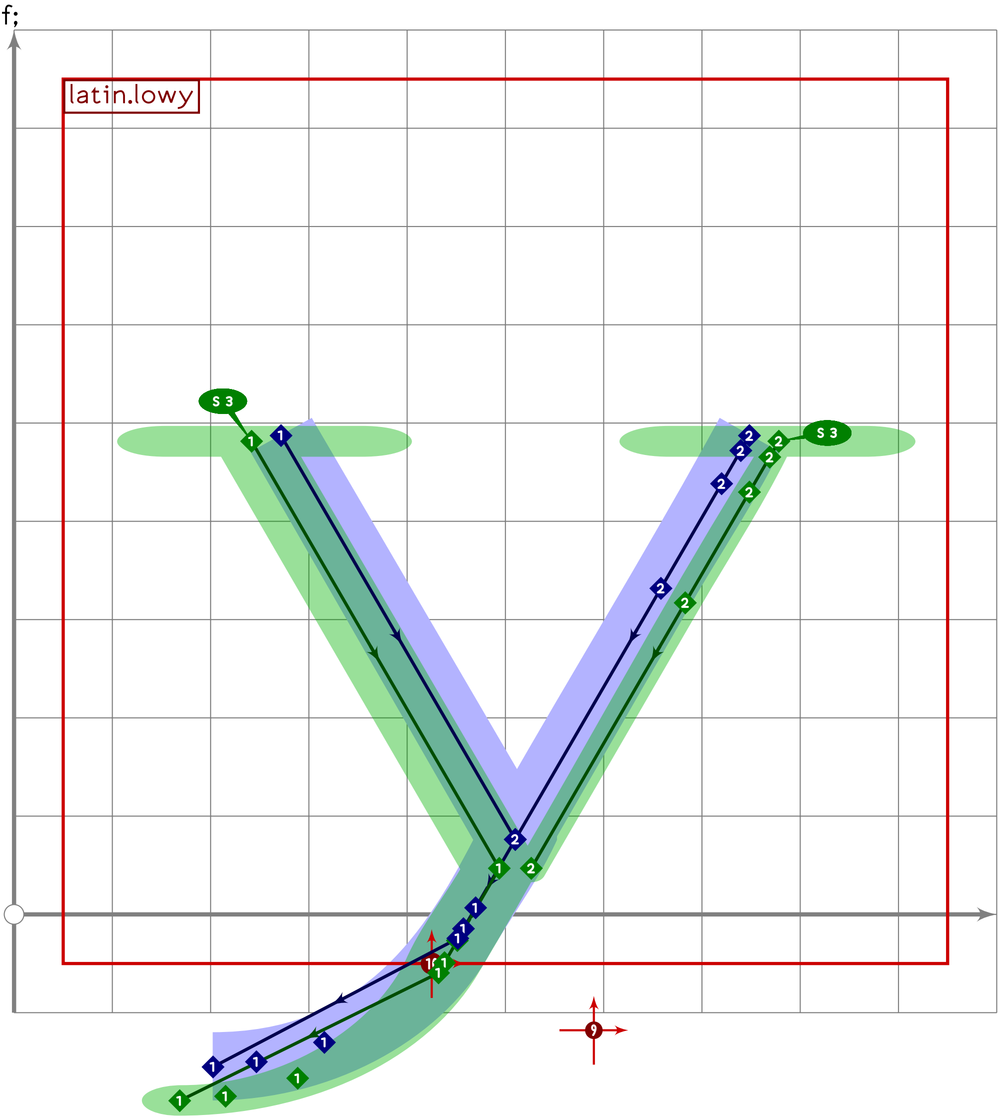
U+FF58
tsuku.uniFF58



```
vardef latin.lowx =
  push_pbox_toexpand("latin.lowx");
  (x1+x3)/2=500;
  (x2+x4)/2=500;
  (x2+x3-x1-x4)=((y1-y2)*0.9)*2;
  (x3-x1)=(x2-x4)*0.93;

  y1=y3=latin_wide_xheight_v;
  y2=y4=latin_wide_low_v;

  push_stroke(z1—z2,(1.6,1.6)—(1.6,1.6));
  set_boserif(0,0,if do_italic_hook: 11 else: 3 fi);
  set_boserif(0,1,if do_italic_hook: 11 else: 3 fi);

  if do_alternation:
    push_stroke(z3—(0.5[z3,z4]+alternate_adjust*right/6)
      —(0.5[z3,z4]+alternate_adjust*left/6)—z4,
      (1.6,1.6)—(1.6,1.6)—(1.6,1.6)—(1.6,1.6));
    set_boserif(0,0,if do_italic_hook: 2 else: 3 fi);
    set_boserif(0,3,if do_italic_hook: 1 else: 3 fi);
```

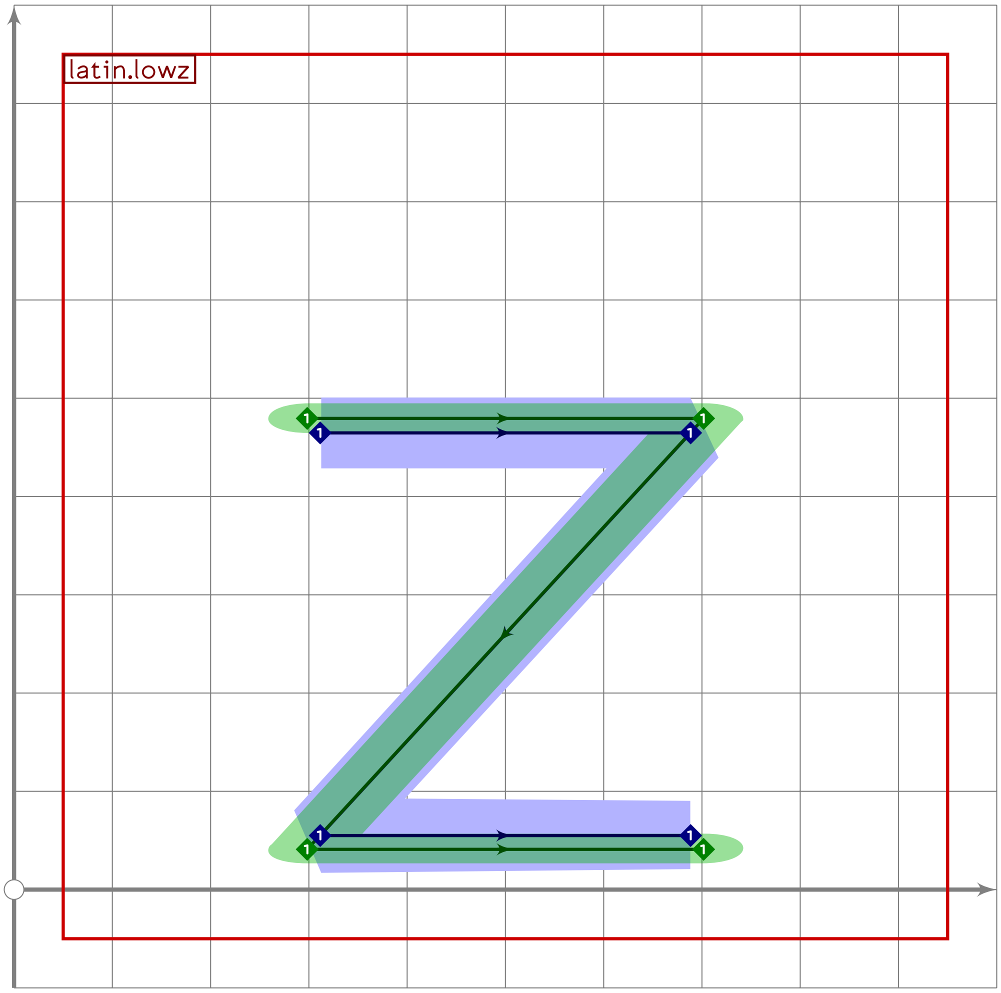
U+FF59
tsuku.uniFF59

```
  else:
    push_stroke(z3−z4,(1.6,1.6)−(1.6,1.6));
    set_boserif(0,0,if do_italic_hook: 2 else: 3 fi);
    set_boserif(0,1,if do_italic_hook: 1 else: 3 fi);
  fi;
  set_boalternate(0);

  tsu_accent.shift_anchors(ai=anc_lower_connect)((240,0));
  expand_pbox;
enddef;
```



```

vardef latin.lowy =
  push_pbox_toexpand("latin.lowy");
  (x1+x3)/2=(x2+x4)/2=510;
  (x2+x3-x1-x4)=((y1-y2)*0.58)*2;
  (x3-x1)=(x2-x4)*0.93;
  x5=x4-0.1*(x2-x4);

  y1=y3=latin_wide_xheight_v;
  y2=y4;
  y5=0.5[y4,latin_wide_low_h]=latin_wide_desc_h;

  push_stroke(z1--z2,(1.6,1.6)--(1.6,1.6));

  push_stroke(z3..tension 10..(0.6[z3,z4])..tension 0.8 and 3..{left}z5,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6));

  numeric xchgtime;
  xchgtime:=ypart (get_strokep(-1) intersectiontimes get_strokep(0));

  replace_strokep(-1)(z1--subpath (xchgtime,infinity) of get_strokep(0));
  replace_strokeq(-1)
    ((1.6,1.6)--subpath (xchgtime,infinity) of get_strokeq(0));

  replace_strokep(0)(subpath (0,xchgtime) of oldp);
  replace_strokeq(0)(subpath (0,xchgtime) of oldq);

  set_boserif(-1,0,if do_italic_hook: 11 else: 3 fi);
  set_boserif(0,0,if do_italic_hook: 2 else: 3 fi);
  set_botip(-1,1,1);
  set_boalternate(0);

  if do_alternation:
    replace_strokep(-1)(oldp shifted (alternate_adjust*left/2));
    replace_strokep(0)(oldp shifted (alternate_adjust*right/2));
  fi;

  tsu_accent.shift_anchors(ai=anc_lower)((90,0));
  tsu_accent.shift_anchors(ai=anc_lower_connect)((-75,-105));
  expand_pbox;
enddef;
```

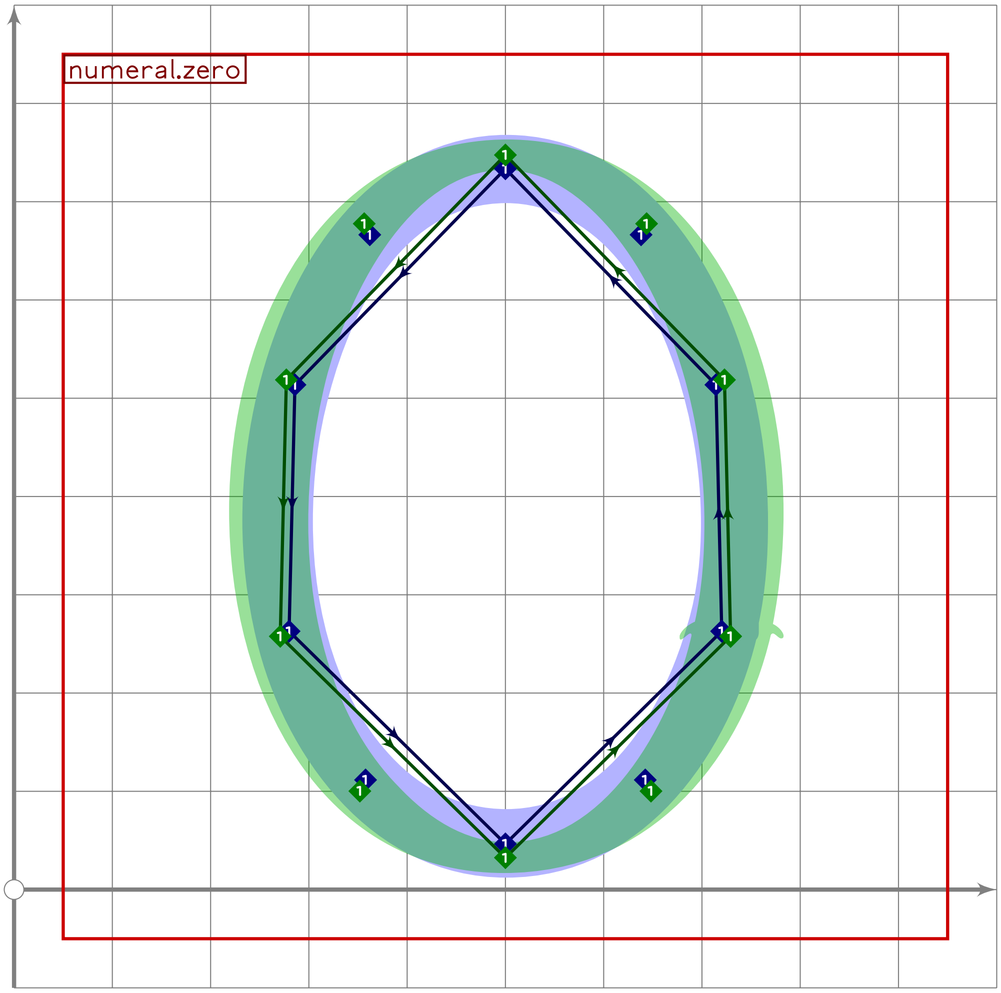
U+FF5A
tsuku.uniFF5A



```
vardef latin.lowz =
  push_pbox_toexpand(”latin.lowz”);
  y1=y2=latin_wide_xheight_h;
  y3=y4=latin_wide_low_h;

  x1=x3;
  x2=x4;
  (x1+x2)/2=500;
  (x2-x1)=(y1-y3)*0.92;

  push_stroke(z1--z2--z3--z4,(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,1,0);
  set_botip(0,2,0);
  expand_pbox;
enddef;
```

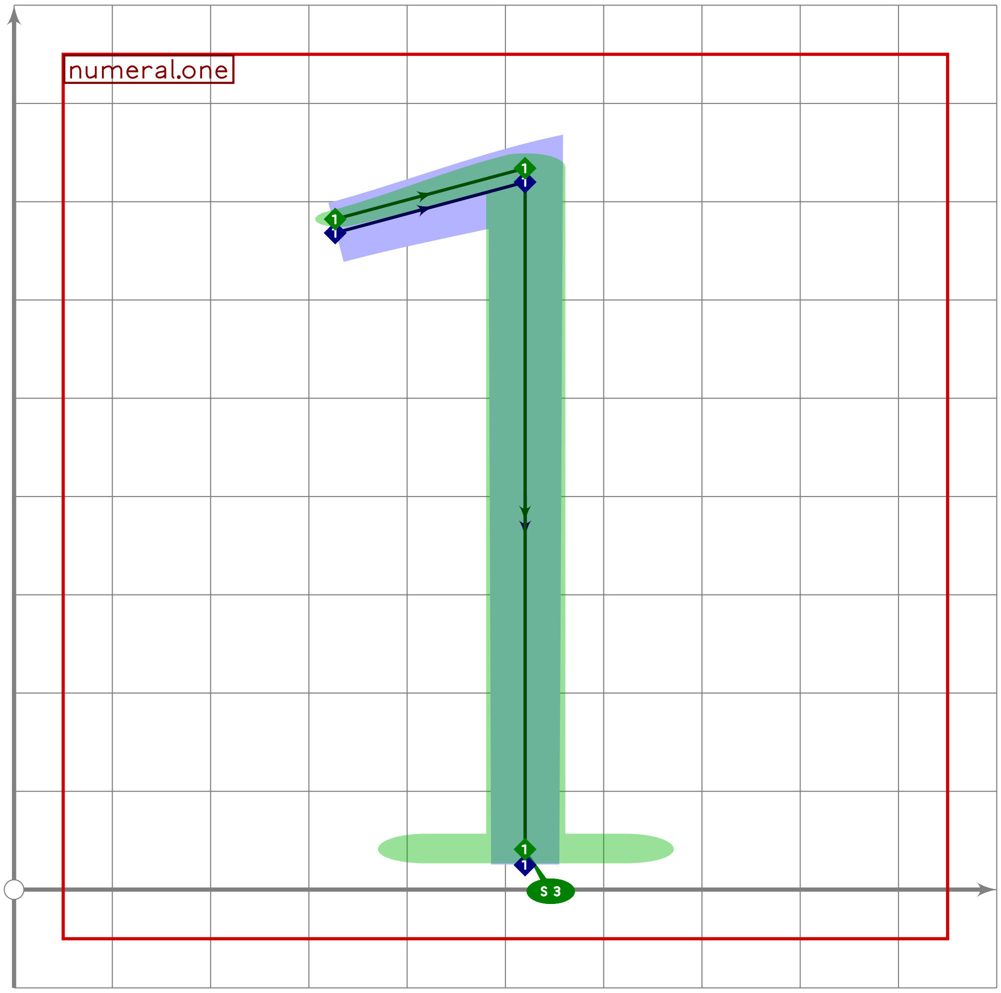
numerals.mp

```
%
% Hindu/Arabic numerals for Tsukurimashou
% Copyright (C) 2011 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(numerals);

```

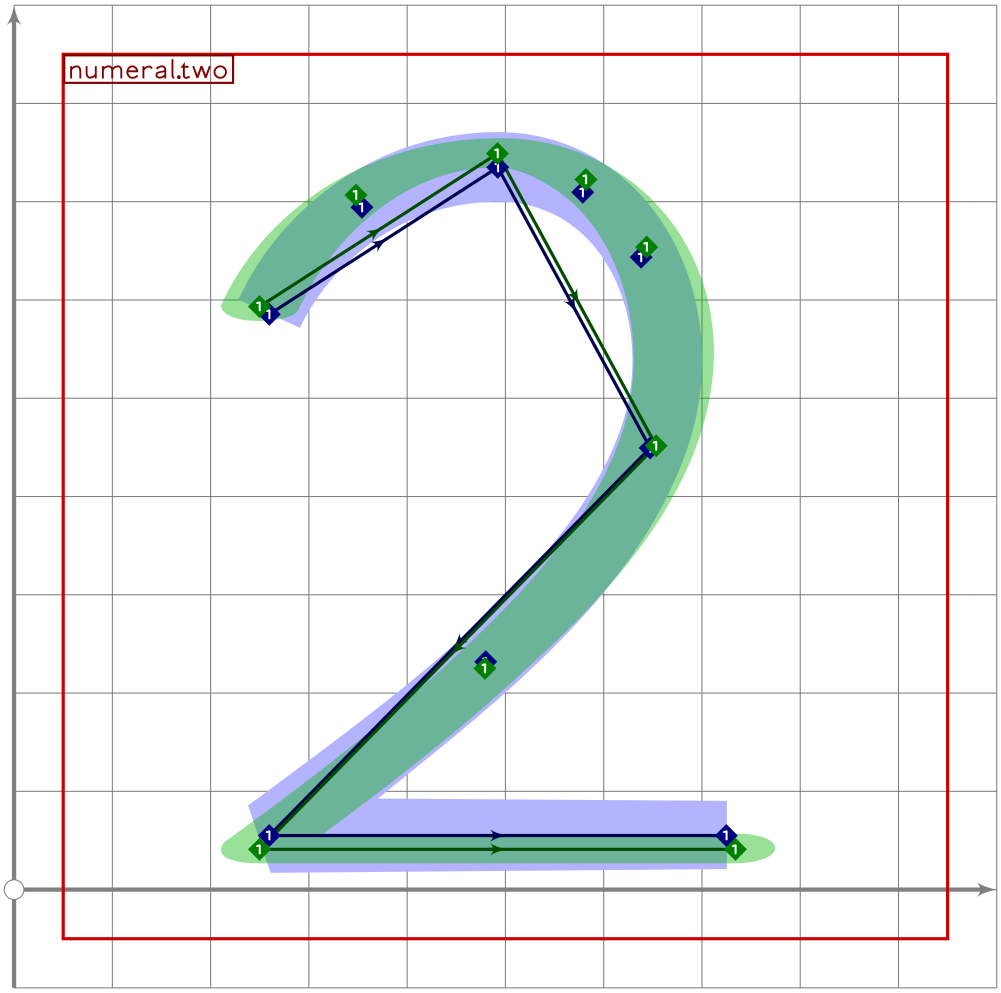


```

vardef numeral.zero =
  push_pbox_toexpand(”numeral.zero”);
  push_stroke(((0.74*dir 330)..(0.72*dir 30)..(up)..
         (0.72*dir 150)..(0.74*dir 210)..(down)..cycle)
    scaled ((latin_wide_high_r-latin_wide_low_r)/2)
    shifted centre_pt,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--cycle);
  expand_pbox;
enddef;
```

U+FF11
tsuku.uniFF11

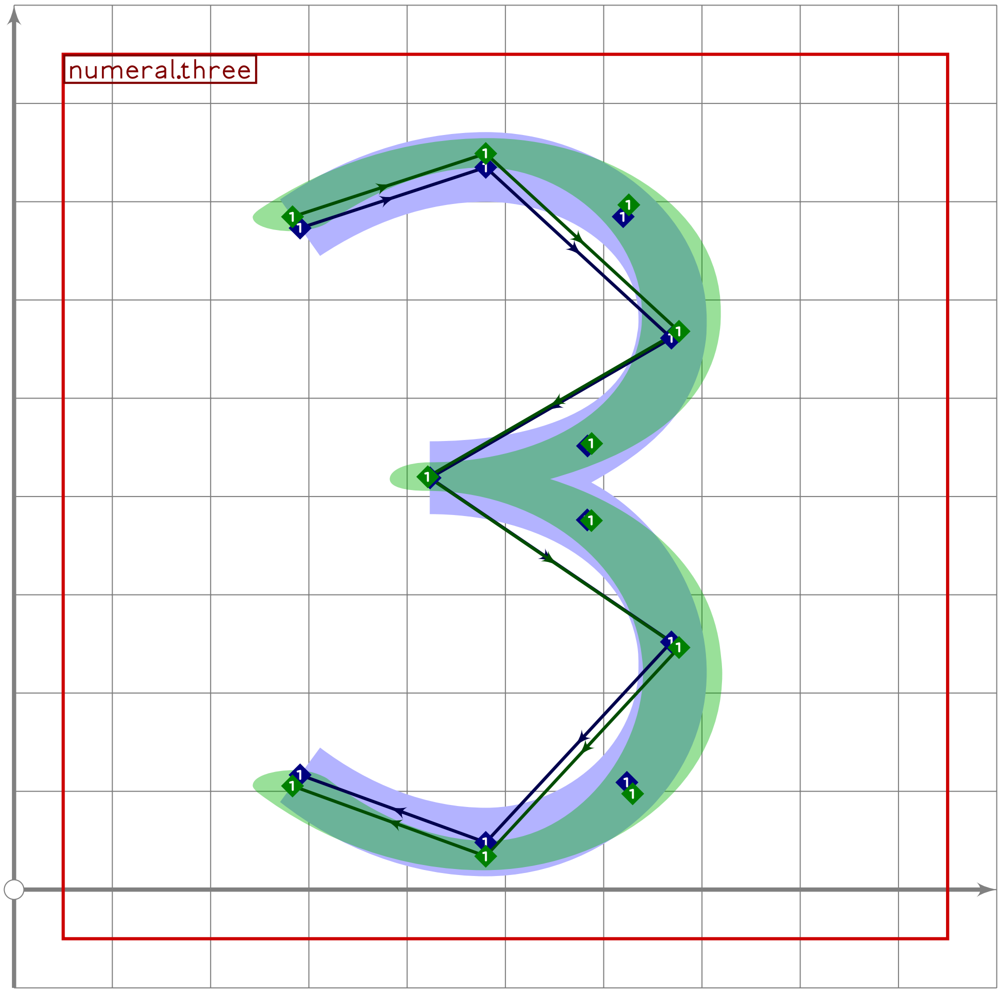


```

vardef  numeral.one  =
  push_pbox_toexpand(”numeral.one”);
  x3=x2=520;

  y2=latin_wide_high_h;
  y3=latin_wide_low_v;

  z1=z2+200*dir  195;

  push_stroke(z1--z2--z3,(1.1,1.1)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,1,1);
  set_boserif(0,2,3);
  expand_pbox;
enddef;
```



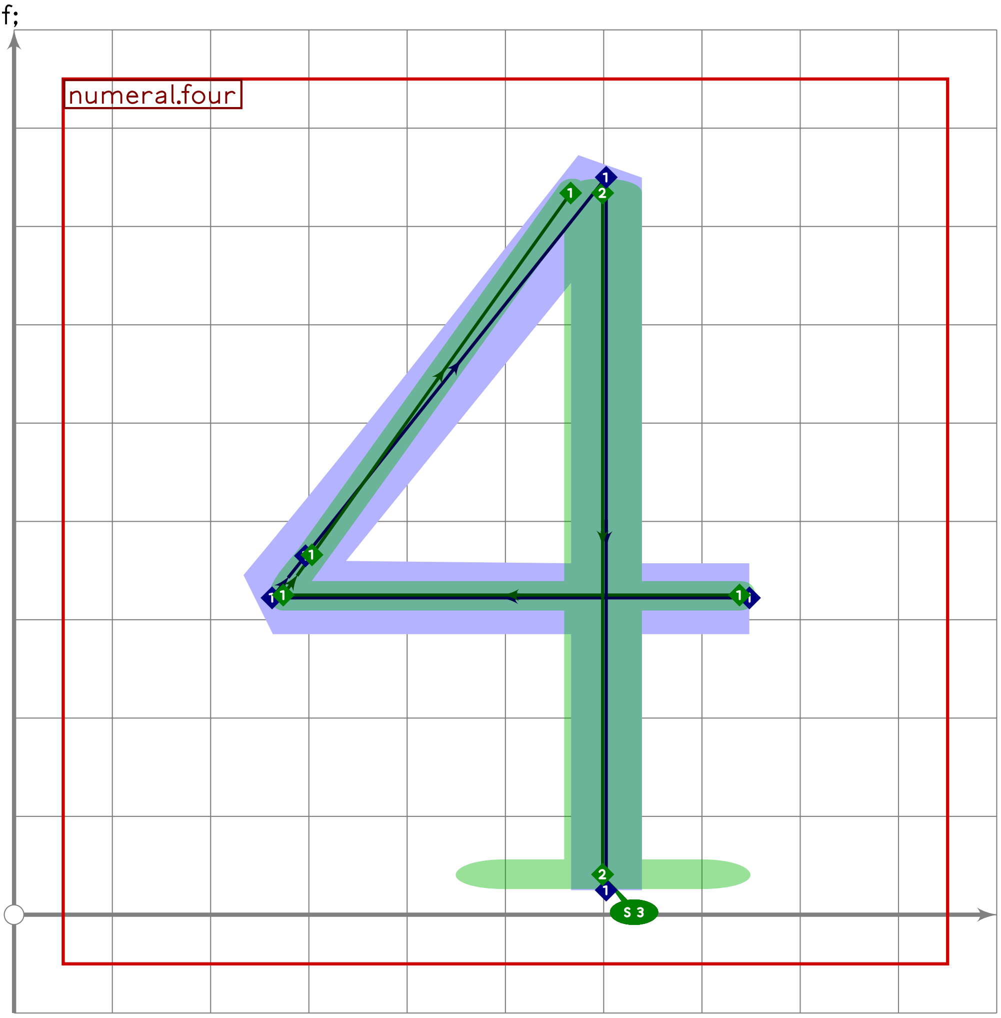
```
vardef numeral.two =
  push_pbox_toexpand(”numeral.two”);
  x1=x4;
  0.62[x1,x3]=500;
  x2=0.6[x1,x3];
  x5=1.2[x1,x3];
  x3-x1=0.57*(y2-y4);

  y1=0.78[y4,y2];
  y2=latin_wide_high_r;
  y3=0.58[y4,y2];
  y4=y5=latin_wide_low_h;

  push_stroke(z1..z2{right}..z3..tension 1.2..{curl 0}z4—z5,
    (1.6,1.6)—(1.6,1.6)—(1.6,1.6)—(1.6,1.6)—(1.6,1.6));
  set_botip(0,3,0);
  expand_pbox;
enddef;
```

U+FF13
tsuku.uniFF13

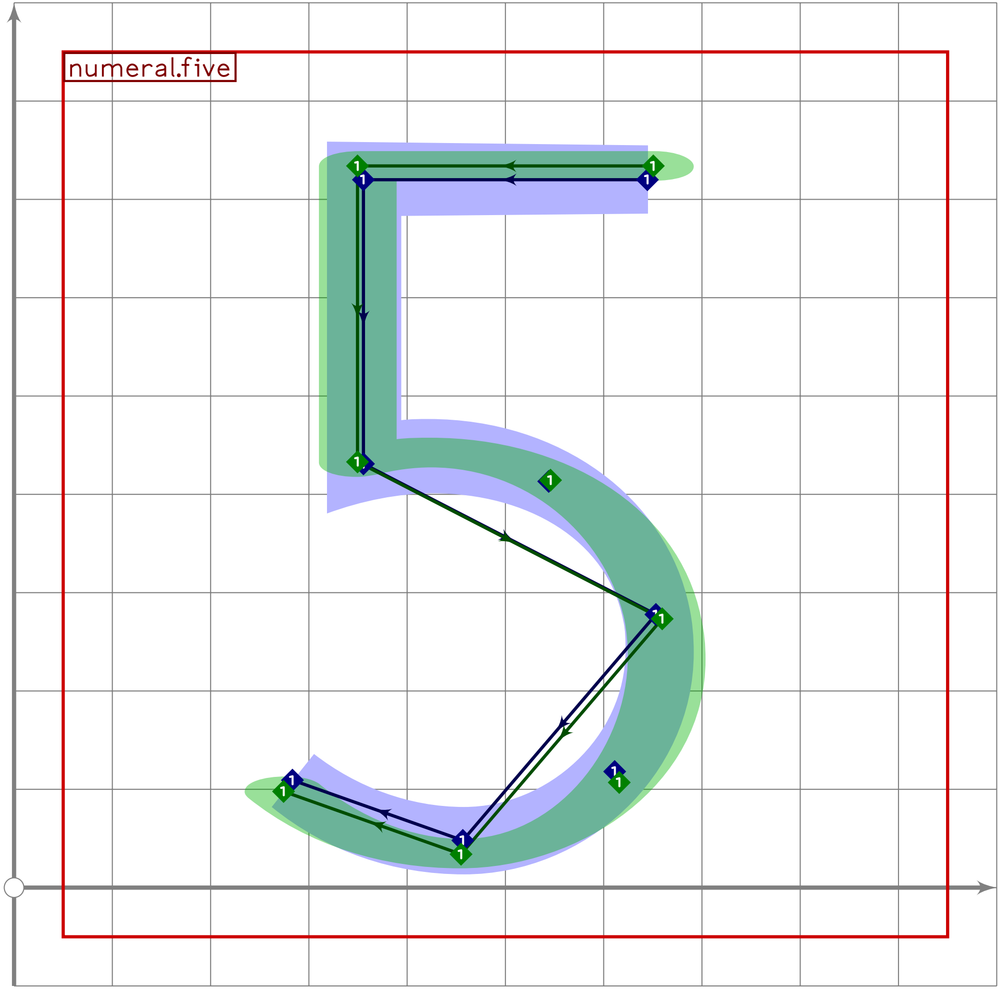


```
vardef numeral.three =
  push_pbox_toexpand(”numeral.three”);
  x1=x7;
  x2=x6=0.5[x1,x3];
  x3=x5;
  x4=0.35[x1,x3];
  (x1+x3)/2=480;
  (x3-x1)=0.55*(y2-y6);

  y1=0.91[y6,y2];
  y2=latin_wide_high_r;
  y3=0.45[y4,y2];
  y4=0.54[y6,y2];
  y5=0.45[y4,y6];
  y6=latin_wide_low_r;
  y7=0.1[y6,y2];

  push_stroke(z1{curl 0.7}..z2{right}..z3..{left}z4{right}..
       z5..z6{left}..{curl 0.7}z7,
    (1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)–
```

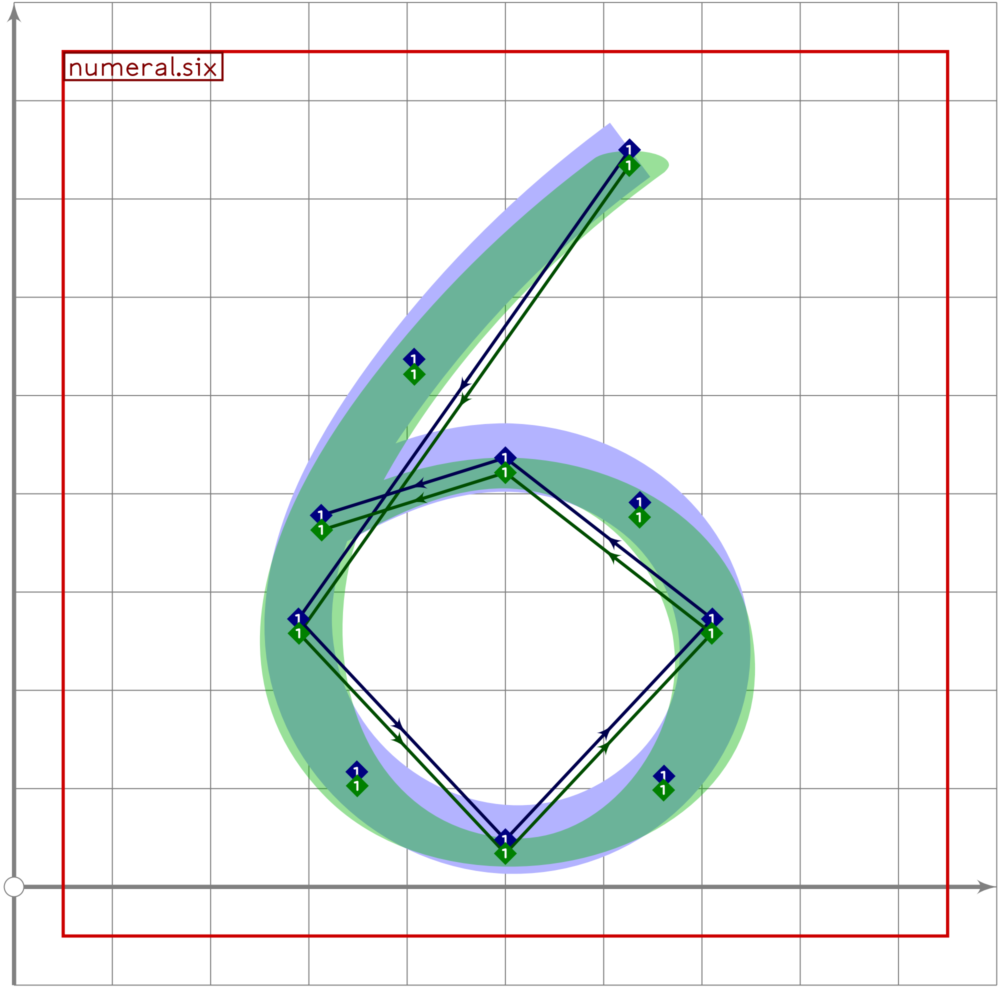
```
      (1.6,1.6)--(1.6,1.6));
  set_botip(0,3,0);
  expand_pbox;
enddef;
```



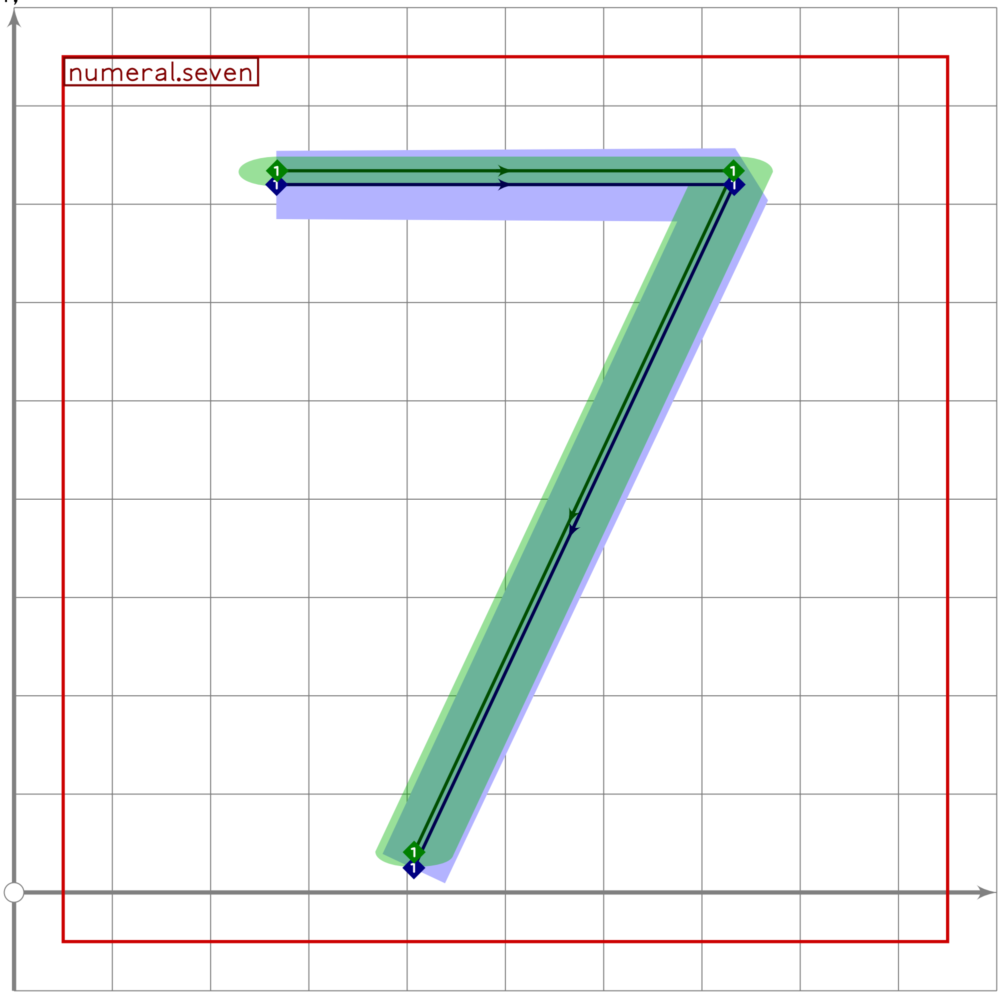
```

vardef numeral.four =
  push_pbox_toexpand(”numeral.four”);
  x3=x4=0.7[x2,x1];
  0.53[x2,x1]=520;
  (x1-x2)=0.67(y3-y4);

  y1=y2=0.41[y4,y3];
  y3=latin_wide_high_v;
  y4=latin_wide_low_v;

  if do_alternation:
    push_stroke(z1--z2--(0.1[z2,(z3+alternate_adjust*left)])--
        (z3+alternate_adjust*left),
      (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
    set_botip(0,1,0);
    set_botip(0,2,0);
    set_boalternate(0);

    push_stroke(z3−z4,(1.6,1.6)−(1.6,1.6));
    set_botip(0,0,0);
    set_boserif(0,1,3);
  else:
    push_stroke(z1−z2−(0.1[z2,z3])−z3−z4,
      (1.6,1.6)−(1.6,1.6)−(1.6,1.6)−(1.6,1.6)−(1.6,1.6));
    set_botip(0,1,0);
    set_botip(0,3,0);
    set_boserif(0,4,3);
  fi;
  expand_pbox;
enddef;
```



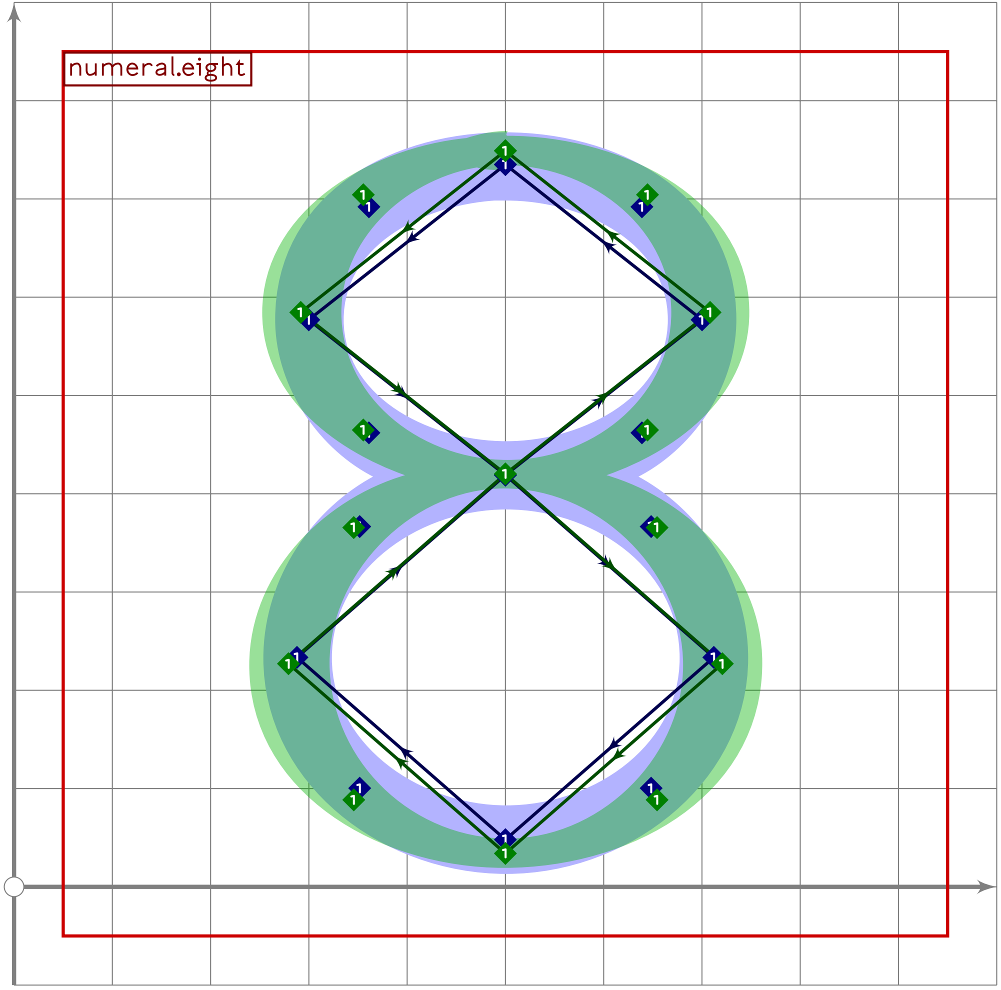
```

vardef numeral.five =
  push_pbox_toexpand("numeral.five");
  (x1+x2)/2=500;
  (x1-x2)=(y2-y3);
  x2=x3;
  x4=1.03[x2,x1];
  x5=0.35[x2,x1];
  x6=(-0.25)[x2,x1];

  y1=y2=latin_wide_high_h;
  y3=0.57[y5,y1];
  y4=0.6[y5,y3];
  y5=latin_wide_low_r;
  y6=0.16[y5,y3];

  push_stroke(z1–z2–z3{curl 0.5}..z4..z5{left}..z6,
    (1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6));
  set_botip(0,1,1);
  set_botip(0,2,1);
  expand_pbox;
enddef;
```



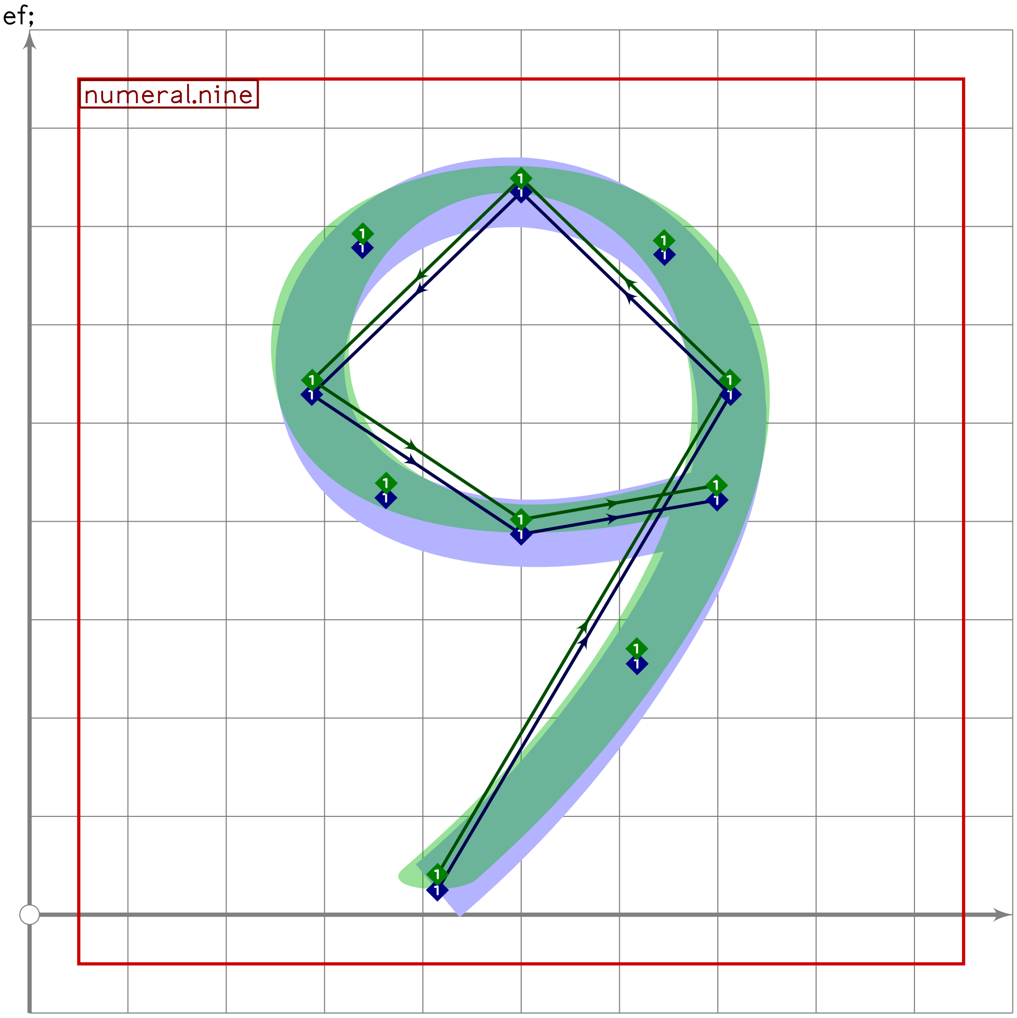
```

vardef numeral.six =
  push_pbox_toexpand(”numeral.six”);
  x1=0.8[x2,x4];
  (x2+x4)/2=x3=500;
  (x4-x2)=0.6(y1-y3);
  x5=x3;

  y1=latin_wide_high_v;
  y2=y4=0.32[y3,y1];
  y3=latin_wide_low_r;
  y5-y4=0.73*(y4-y3);

  push_stroke(z1{curl 0.2}..tension 1.2..z2..z3..z4{dir 100},
    (1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6));
  replace_strokep(0)(z1{curl 0.2}..tension 1.2..z2..z3..z4{dir 100}..
    z5..{curl 0.2}(point 0.8 of oldp));
  expand_pbox;
enddef;
```



```

vardef numeral.seven =
  push_pbox_toexpand("numeral.seven");
  (x1+x2)/2=500;
  x3=0.3[x1,x2];
  (x2-x1)=0.67*(y1-y3);

  y1=y2=latin_wide_high_h;
  y3=latin_wide_low_v;

  push_stroke(z1--z2--z3,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_botip(0,1,0);
  expand_pbox;
enddef;
```



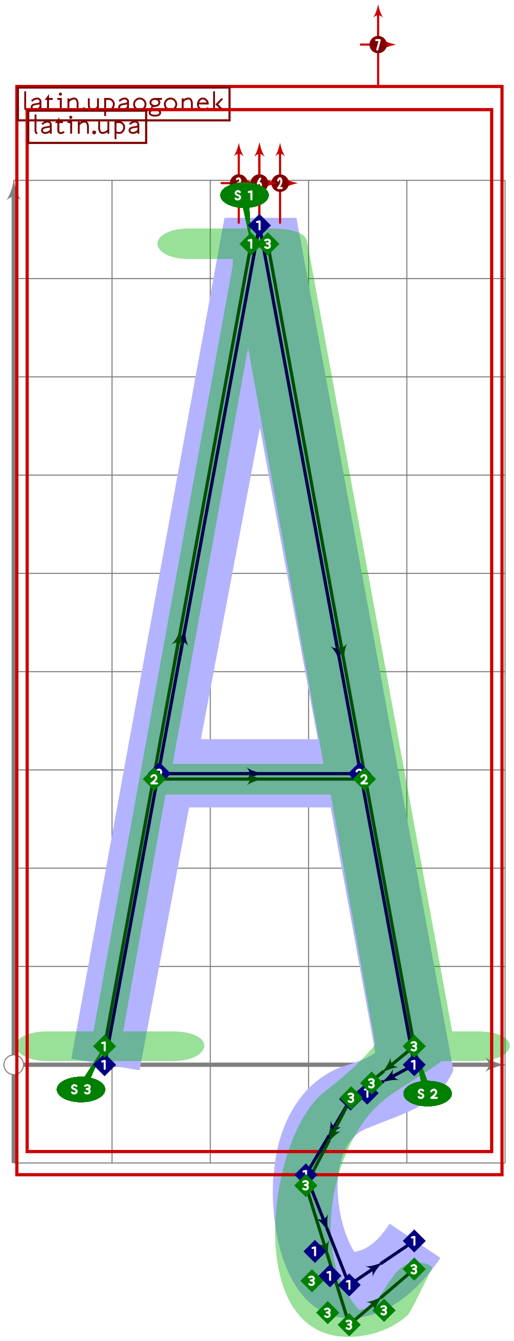
```

vardef  numeral.eight =
  push_pbox_toexpand("numeral.eight");
  x1=x3=x5=x7=(x2+x8)/2=(x4+x6)/2=500;
  (x4-x6)=1.06*(x8-x2);
  (x4+x8-x6-x2)/2=0.6*(y1-y5);

  y1=latin_wide_high_r;
  y2=y8=0.5[y3,y1];
  y3=y7=0.54[y5,y1];
  y4=y6=0.5[y5,y3];
  y5=latin_wide_low_r;

  push_stroke(z1..z2..z3{right}..z4..z5..z6..z7{right}..z8..cycle,
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--
      (1.6,1.6)--(1.6,1.6)--cycle);
  expand_pbox;
enddef;
```



```

vardef numeral.nine =
  push_pbox_toexpand("numeral.nine");
  x1=0.3[x4,x2];
  (x2+x4)/2=x3=500;
  (x2-x4)=0.6(y3-y1);
  x5=x3;

  y1=latin_wide_low_v;
  y2=y4=0.29[y3,y1];
  y3=latin_wide_high_r;
  y5-y4=0.69*(y4-y3);

  push_stroke(z1{curl 0.2}..tension 1.2..z2..z3..z4{dir 280},
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(z1{curl 0.2}..tension 1.2..z2..z3..z4{dir 280}..
    z5..{curl 0.2}(point 0.8 of oldp));
  expand_pbox;
enddef;
```

ogonek.mp

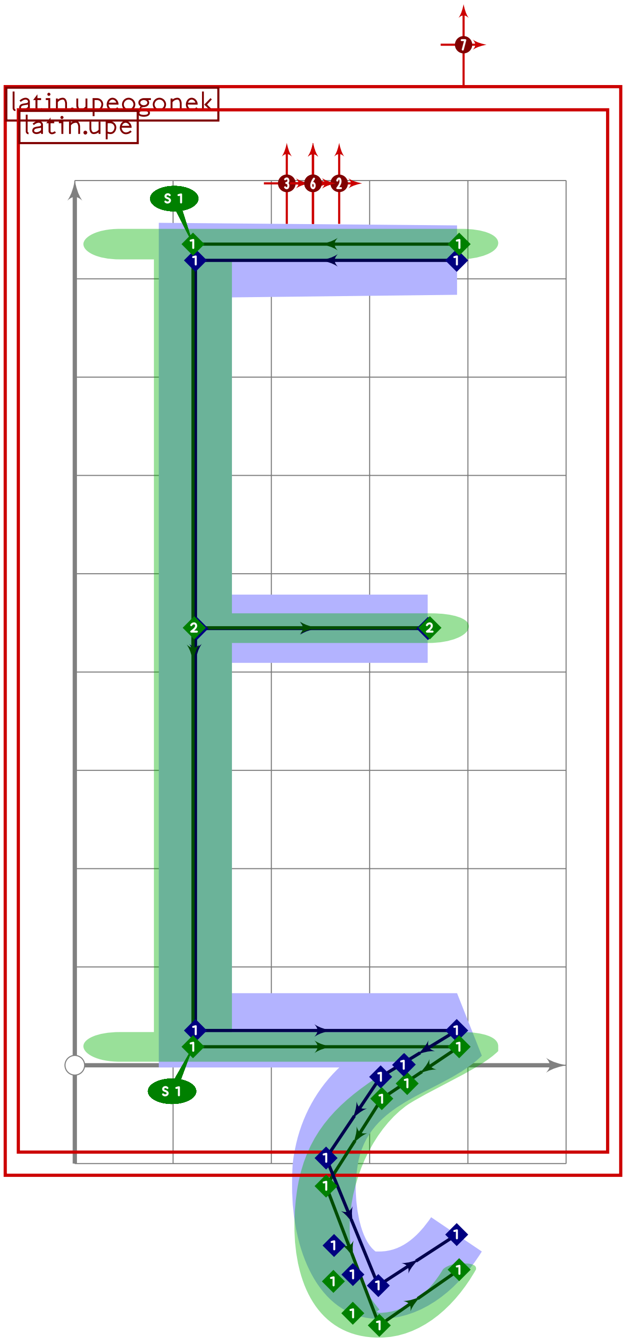
```
%
% Ogonek letters for Tsukurimashou
% Copyright (C) 2011, 2012 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(ogonek);

```



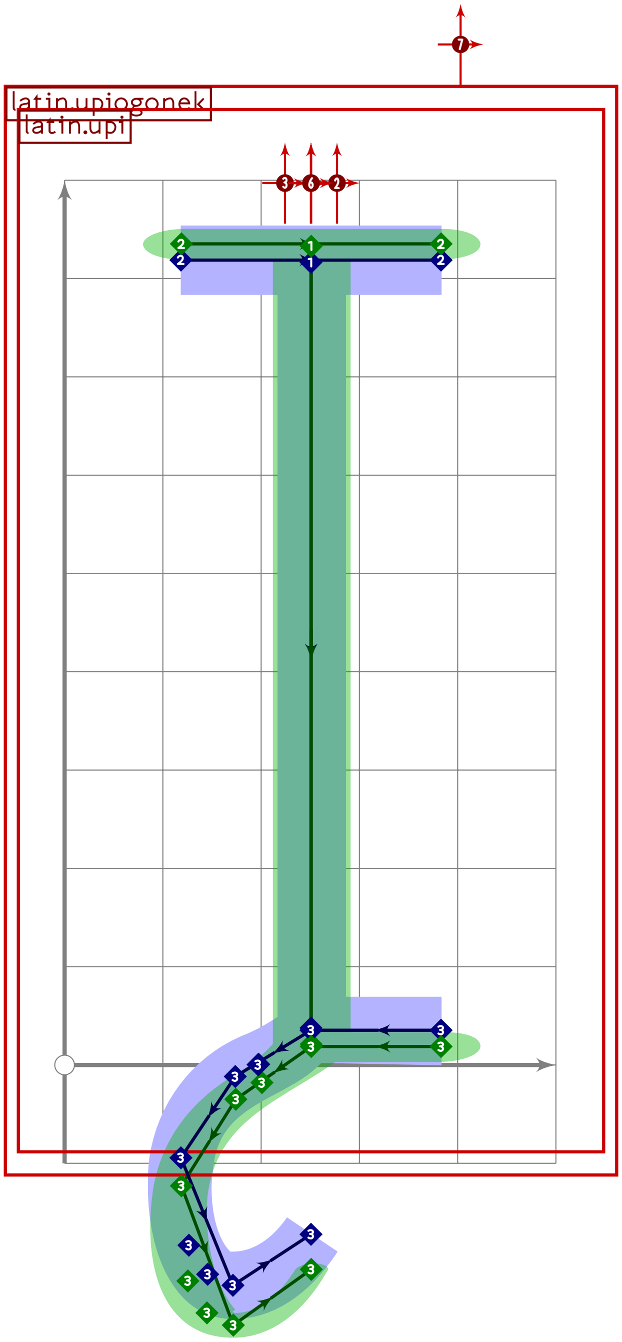
OGON

```
vardef latin.upaogonek =
  push_pbox_toexpand(”latin.upaogonek”);
  latin.upa;

  x6=0.3[x2,x3];
  x7=0.4[x6,x8];
  x8=x3;

  y6=0.5[y7,y3];
  y7=latin_wide_desc_r;
  y8=0.2[y7,y3];

  if do_alternation:
    replace_strokep(0)(oldp{dir 210}..z6..z7{right}..z8);
    replace_strokep(0)(insert_nodes(oldp)(length(oldp)-2.5));
    replace_strokeq(0)(oldq–(1.4,1.4)–(1.3,1.3)–(1.4,1.4)–(1,1));
    set_boserif(0,1,2);
    set_botip(0,length(get_strokep(0))-4,1);
  else:
    replace_strokep(-1)(oldp{dir 210}..z6..z7{right}..z8);
    replace_strokep(-1)(insert_nodes(oldp)(length(oldp)-2.5));
    replace_strokeq(-1)(oldq–(1.4,1.4)–(1.3,1.3)–(1.4,1.4)–(1,1));
    set_boserif(-1,1,2);
    set_botip(-1,length(get_strokep(-1))-4,1);
  fi;
  expand_pbox;
enddef;
```

U+0118
tsuku.Eogonek

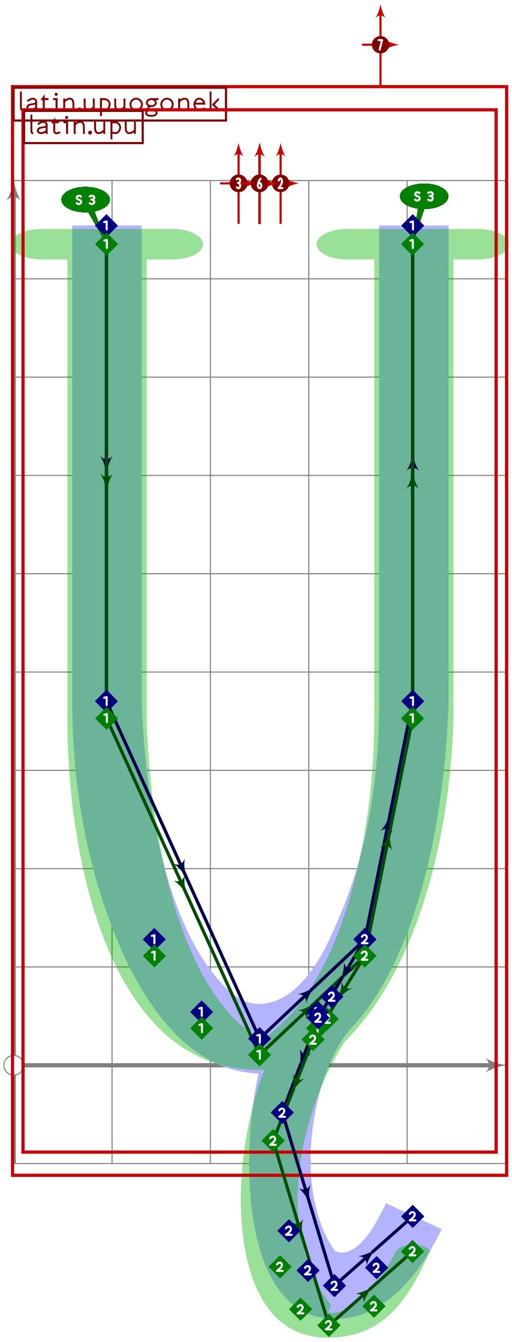


```
vardef latin.upeogonek =
  push_pbox_toexpand(”latin.upeogonek”);
  latin.upe;

  x7=0.5[x3,x4];
  x8=0.4[x7,x9];
  x9=x4;

  y7=0.5[y8,y4];
  y8=latin_wide_desc_r;
  y9=0.2[y8,y4];

  replace_strokep(-1)(oldp{dir 210}..z7..z8{right}..z9);
  replace_strokep(-1)(insert_nodes(oldp)(length(oldp)-2.5));
  replace_strokeq(-1)(oldq--(1.4,1.4)--(1.3,1.3)--(1.4,1.4)--(1,1));
  set_botip(-1,length(get_strokep(-1))-4,0);
  expand_pbox;
enddef;
```



```

vardef latin.upiogonek =
  push_pbox_toexpand(”latin.upiogonek”);
  latin.upi;

  x1=x4=500;
  x2=300;
  x3=0.4[x2,x4];

  y1=latin_wide_low_h;
  y2=0.5[y3,y1];
  y3=latin_wide_desc_r;
  y4=0.2[y3,y1];

  replace_strokep(0)((700,latin_wide_low_h)--z1{dir 210}..z2..z3{right}..z4);
  replace_strokep(0)(insert_nodes(oldp)(1.5));
  replace_strokeq(0)((1.6,1.6)--(1.6,1.6)--(1.4,1.4)--
    (1.3,1.3)--(1.4,1.4)--(1,1));
  expand_pbox;
enddef;
```

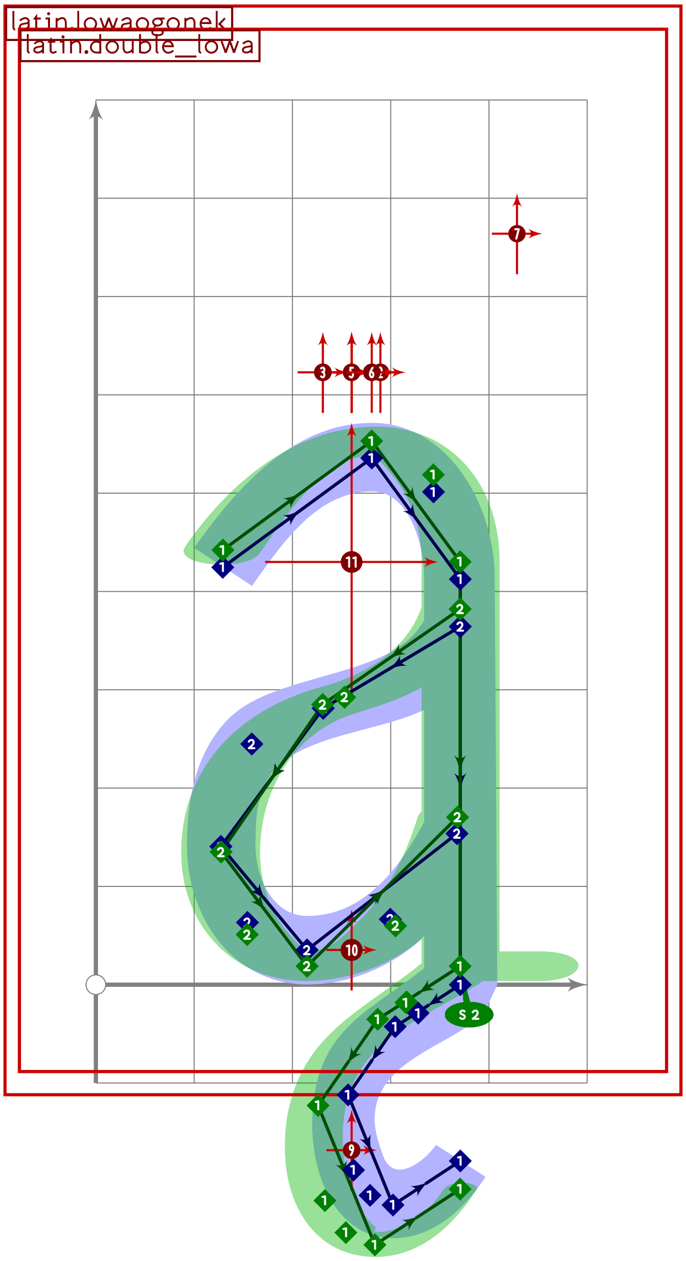
U+0172
tsuku.Uogonek



```

vardef latin.upuogonek =
  push_pbox_toexpand("latin.upuogonek");
  latin.upu;

  replace_strokep(0)(insert_nodes(oldp)(2.5));
  replace_strokeq(0)(insert_nodes(oldq)(2.5));
  set_boserif(0,5,3);
  set_boserif(0,4,whatever);

  z6=point 3 of get_strokep(0);

```

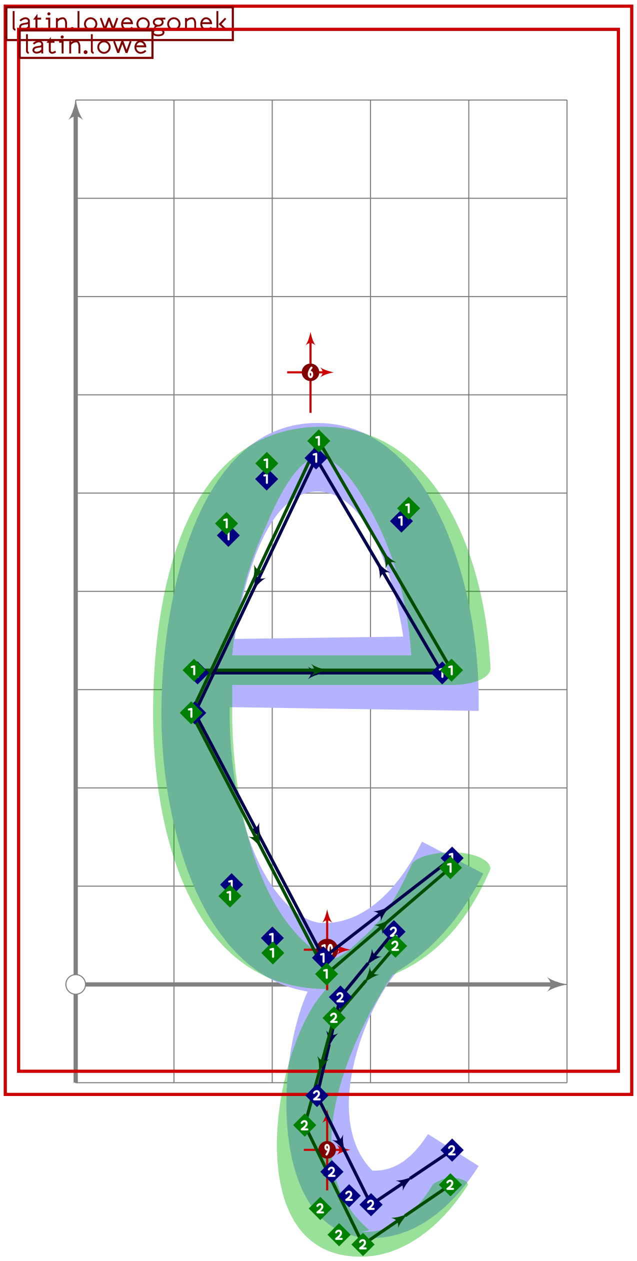
U+0105
tsuku.aogonek

```
    y7=0.5[y8,y6];
    y8=latin_wide_desc_r;
    y9=0.2[y8,y6];

    x9-x7=(x4-x2)*((y6-y8)/(y1-y3));
    x8=0.4[x7,x9];
    x9=x4;

    push_stroke(z6{-direction 3 of get_strokep(0)}..z7..z8{right}..z9,
      (1.1,1.1)--(1.4,1.4)--(1.3,1.3)--(1.4,1.4)--(1,1));
    replace_strokep(0,insert_nodes(oldp)(length(oldp)-2.5));
    expand_pbox;
enddef;
```

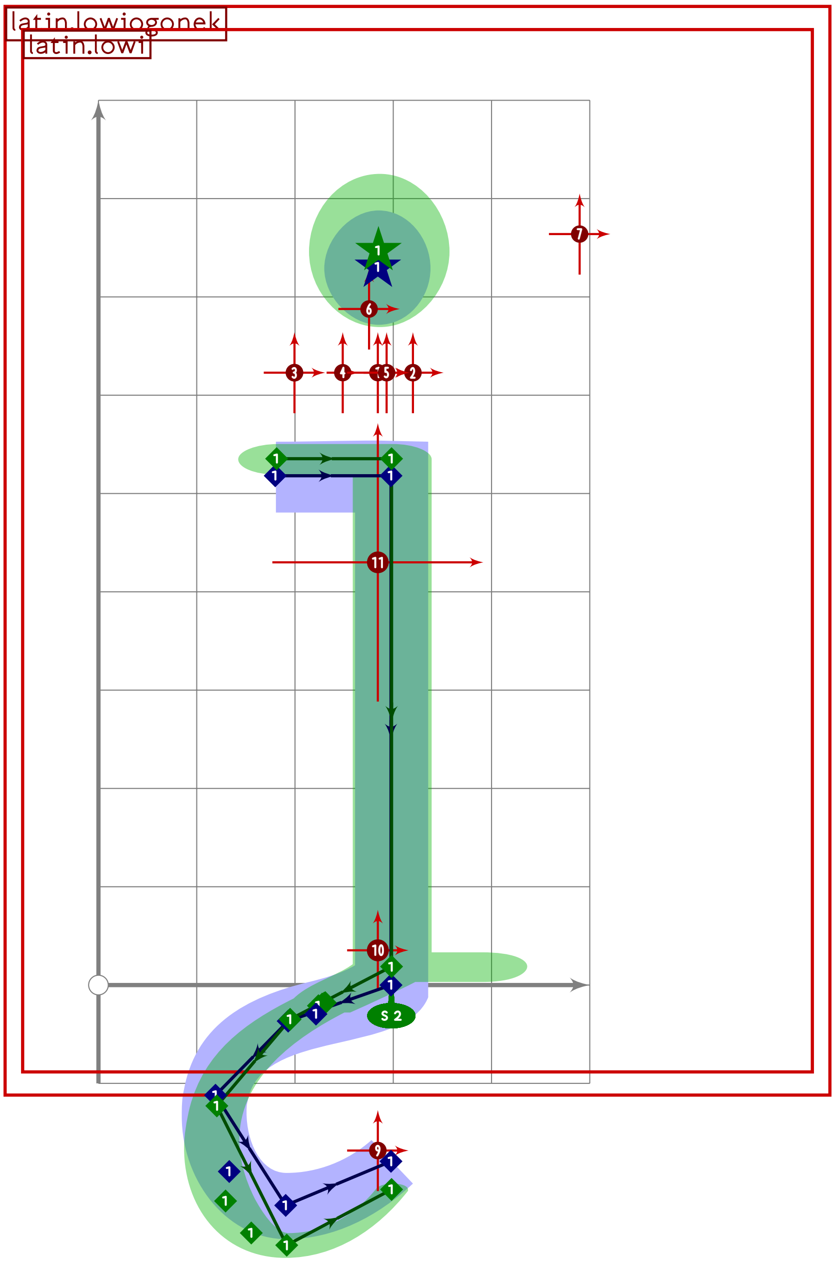


```
vardef latin.lowaogonek =
  push_pbox_toexpand(”latin.lowaogonek”);
  latin.lowa;

  y9=0.5[y10,y4];
  y10=latin_wide_desc_r;
  y11=0.2[y10,y4];

  x11-x9=(x4-x1)*((y4-y10)/(y2-y4));
  x10=0.4[x9,x11];
  x11=x4;

  replace_strokep(-1)(oldp{dir 210}..z9..z10{right}..z11);
  replace_strokep(-1)(insert_nodes(oldp)(length(oldp)-2.5));
  replace_strokeq(-1)(oldq--(1.4,1.4)--(1.3,1.3)--(1.4,1.4)--(1,1));
  set_botip(-1,length(get_strokep(-1))-4,1);
  expand_pbox;
enddef;
```

U+0119
tsuku.eogonek



```
vardef latin.loweogonek =
  push_pbox_toexpand("latin.loweogonek");
  latin.lowe;

  z7=point (-0.5+length get_strokep(0)) of get_strokep(0);

  y8=0.4[y9,y7];
  y9=latin_wide_desc_r;
  y10=0.2[y9,y7];

  x10-x8=(x2-x4)*((y7-y9)/(y3-y5));
  x9=0.4[x8,x10];
  x10=x6;

  push_stroke(z7{-direction (-0.5+length get_strokep(0)) of get_strokep(0)}
      ..z8..z9{right}..z10,
    (1.1,1.1)–(1.4,1.4)–(1.3,1.3)–(1.4,1.4)–(1,1));
  replace_strokep(0,insert_nodes(oldp)(length(oldp)-2.5));
  expand_pbox;
enddef;
```

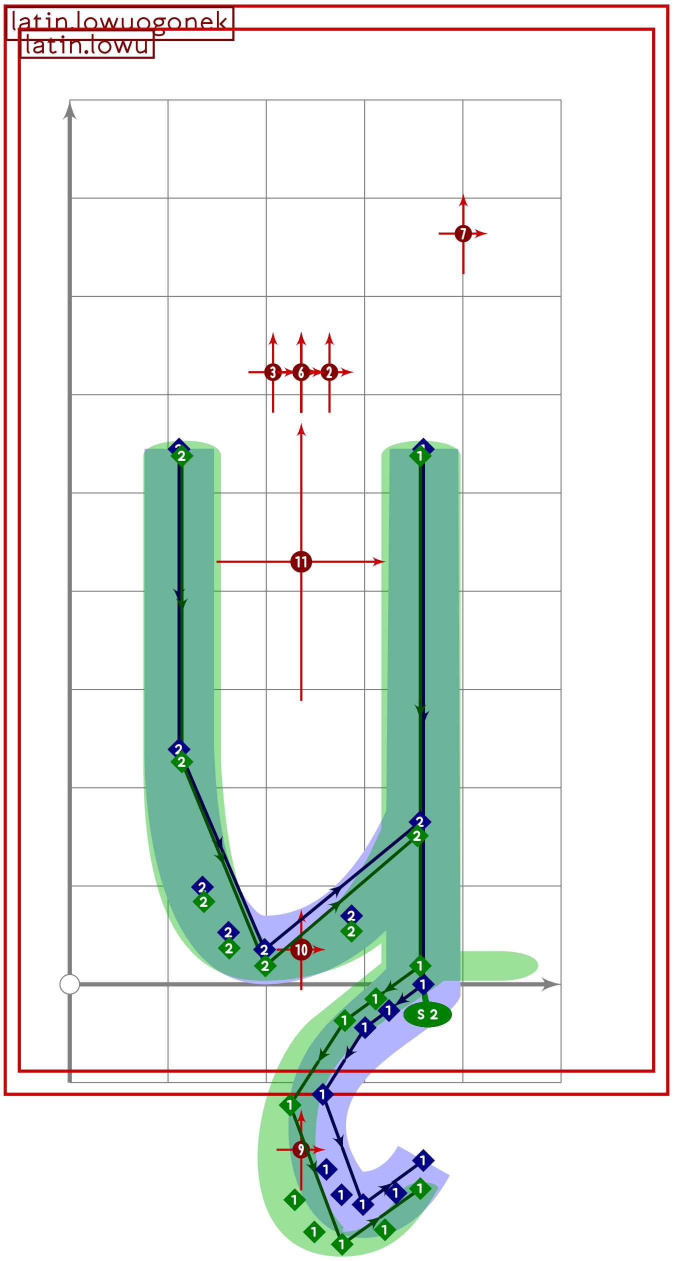


```

vardef latin.lowiogonek =
  push_pbox_toexpand("latin.lowiogonek");
  latin.lowi;

  x5=x7-200;
  x6=0.4[x5,x7];
```

OGON

U+0173
tsuku.uogonek

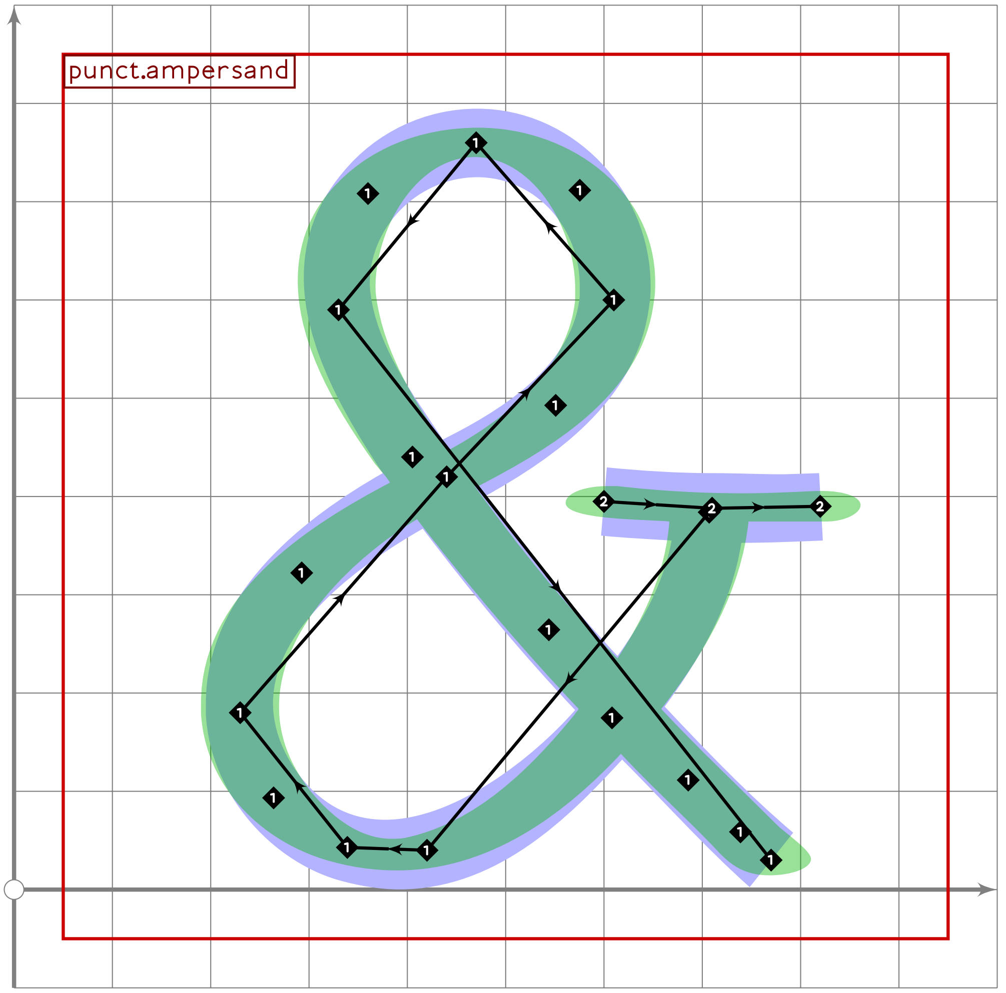
```
  x7=x3;

  y5=0.5[y6,y3];
  y6=latin_wide_desc_r;
  y7=0.2[y6,y3];

  replace_strokep(0)(oldp{dir 210}..z5..z6{right}..z7);
  replace_strokep(0)(insert_nodes(oldp)(2.5));
  replace_strokeq(0)(oldq–(1.4,1.4)–(1.3,1.3)–(1.4,1.4)–(1,1));
  set_boserif(0,2,2);
  set_botip(0,2,1);
  expand_pbox;
enddef;
```



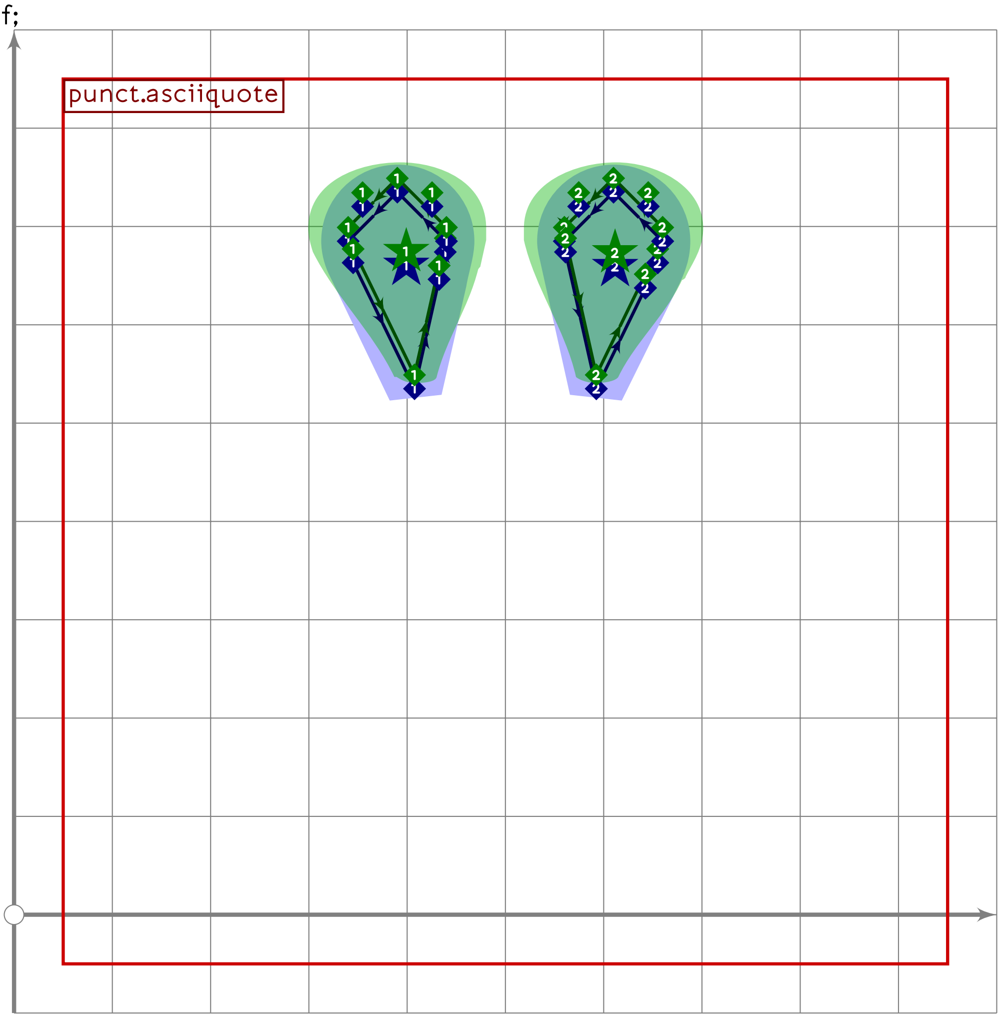
```
vardef latin.lowuogonek =
  push_pbox_toexpand("latin.lowuogonek");
  latin.lowu;

  y7=0.5[y8,y1];
  y8=latin_wide_desc_r;
  y9=0.2[y8,y1];

  x9-x7=(x1-x6)*((y1-y8)/(y2-y1));
  x8=0.4[x7,x9];
  x9=x1;

  replace_strokep(-1)(z2--z1{dir 210}..z7..z8{right}..z9);
  replace_strokep(-1)(insert_nodes(oldp)(length(oldp)-2.5));
  replace_strokeq(-1)((1.6,1.6)--(1.6,1.6)--(1.4,1.4)--(1.3,1.3)--
    (1.4,1.4)--(1,1));
  set_botip(-1,1,1);
  set_boserif(-1,0,whatever);
  set_boserif(-1,1,2);
  expand_pbox;
enddef;
```

punct.mp

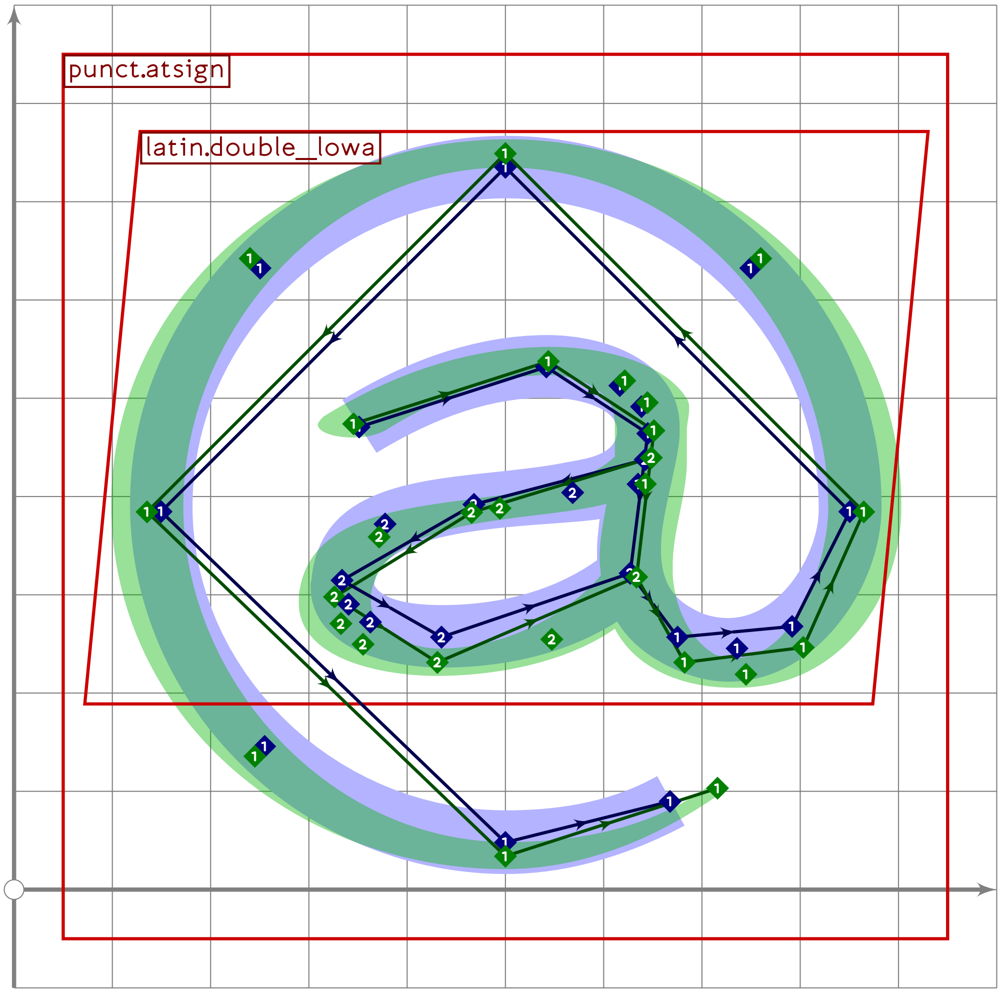
```
%
% Punctuation for Tsukurimashou
% Copyright (C) 2011 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(punct);

```



```

vardef punct.ampersand =
  push_pbox_toexpand("punct.ampersand");

  push_stroke((707,384)..tension 1.3..(420,40)..(230,180)..(440,420)..
      (610,600)..(470,760)..(330,590)..tension 1.5 and 4..(770,30),
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--
      (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  replace_strokep(0)(insert_nodes(oldp)(1.3));

  push_stroke((600,395)..(710,388)..(820,390),
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  expand_pbox;
enddef;
```



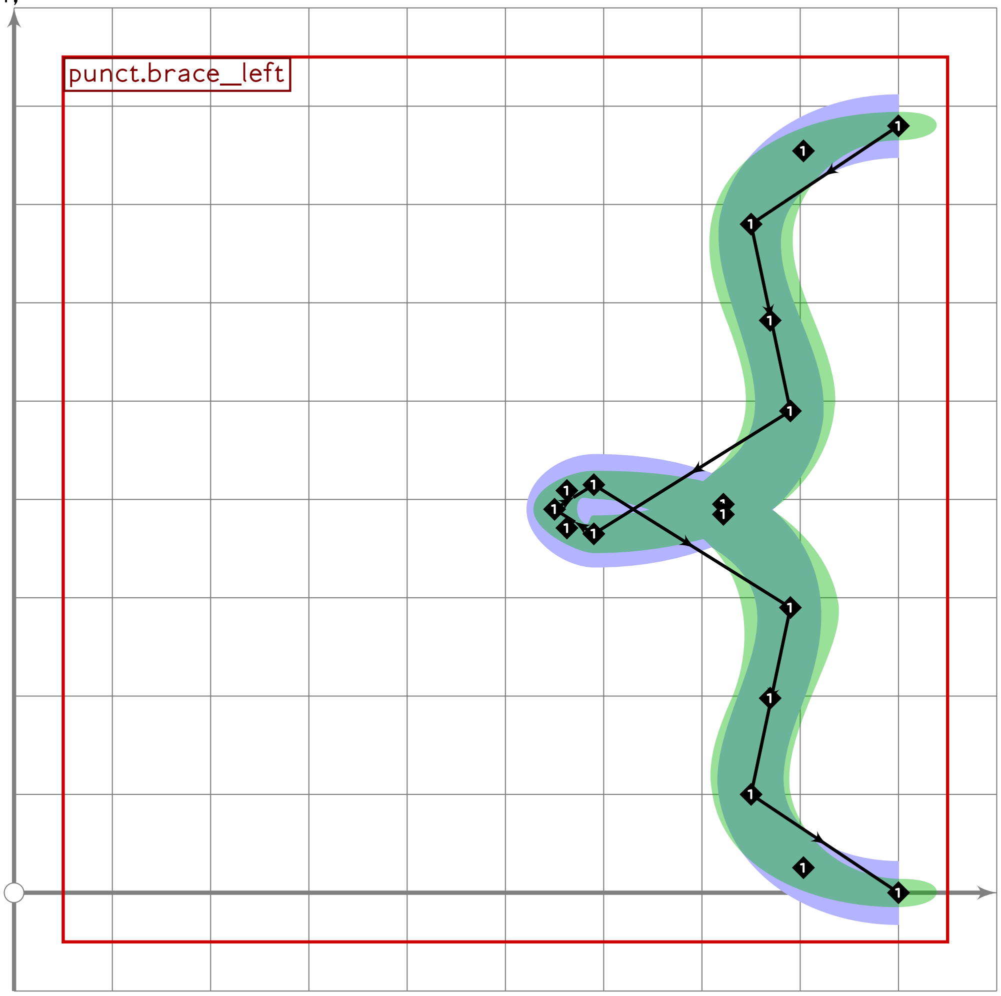
```

vardef punct.asciiquote =
  push_pbox_toexpand("punct.asciiquote");

  numeric dx;
  dx=tsu_punct_size;

  (x1+x2)/2=(x3+x4)/2=500;
  x2-x1=1.2*dx+tsu_punct_size;
  x4-x3=tsu_punct_size*1.85;

  y1=y2=latin_wide_high_r-dx/2;
  y3=y4=y1-1.5*dx;

  path ptmp;
  ptmp:=(down..right..up..left..cycle)
    scaled (abs(z3-z1)+-+(dx/2));

  push_stroke((down..right..up..left..cycle) scaled (dx/2) shifted z1,
    (2,2)--(2,2)--(2,2)--(2,2)--(2,2)--(2,2)--(1.3,1.3)--cycle);
  replace_strokep(0)(z3--(subpath ((xpart (oldp intersectiontimes
    (ptmp shifted z3))),(4-xpart ((reverse oldp) intersectiontimes
    (ptmp shifted z3)))) of oldp)--cycle);
  replace_strokep(0)((subpath (0.8,6) of oldp)--cycle);
  set_bosize(0,75);
  set_botip(0,6,0);;

  push_stroke((down..right..up..left..cycle) scaled (dx/2) shifted z2,
    (2,2)--(2,2)--(2,2)--(2,2)--(2,2)--(2,2)--(1.3,1.3)--cycle);
  replace_strokep(0)(z4--(subpath ((xpart (oldp intersectiontimes
    (ptmp shifted z4))),(4-xpart ((reverse oldp) intersectiontimes
    (ptmp shifted z4)))) of oldp)--cycle);
  replace_strokep(0)((subpath (0.8,6) of oldp)--cycle);
  set_bosize(0,75);
  set_botip(0,6,0);;

  if tsu_pbrush_size>=30:
    push_lcblob(get_strokep(-1));
    push_lcblob(get_strokep(0));
  fi;
  expand_pbox;
enddef;
```



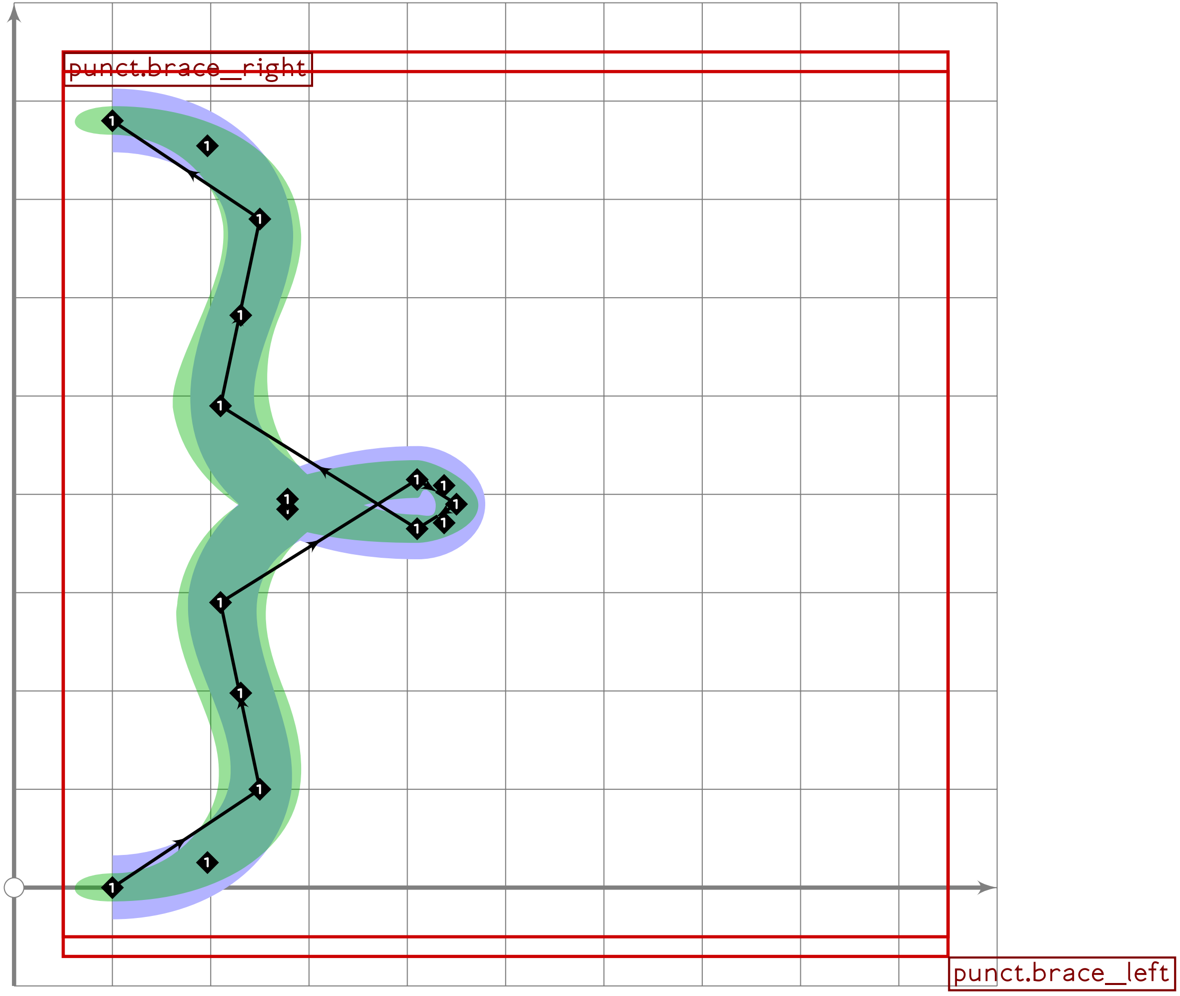
```
vardef punct.atsign =
  push_pbox_toexpand(”punct.atsign”);
  begingroup
    save xsp,ysp;
    xsp:=sp;
    latin.lowa;
    set_boserif(-1,3,whatever);
    ysp:=sp;

    numeric x[],y[];
    x1-x2=x2-x3=y2-y1;
    x2=x4=500;
    y1=y3=0.49[y4,y2];
    y2=latin_wide_high_r;
    y4=latin_wide_low_r;

    transform shrinka;
    (0.5[llcorner get_strokep(-1),urcorner get_strokep(-1)])
      transformed shrinka=0.5[z3,z1];
    (0.5[lrcorner get_strokep(-1),urcorner get_strokep(-1)])
```

U+FF5B
tsuku.uniFF5B

```
      transformed shrinka=0.71[z3,z1];
    (0.5[ulcorner get_strokep(-1),urcorner get_strokep(-1)])
      transformed shrinka=z2+(0.07,-1)*0.29*(x1-x3);
    sp:=xsp;
    tsu_xform(shrinka shifted (-10,0))(sp:=ysp);

    z5=point infinity of get_strokep(0);
    y6=ypart lrcorner get_strokep(0);
    x6=0.5[x2,x1];
    replace_strokep(-1)((subpath (0,length(oldp)-1) of oldp)..z5..z6..
       (subpath (0,3.85) of (z1..z2..z3..z4..cycle)));
    replace_strokep(-1)(insert_nodes(oldp)((length oldp-4.5)));
    replace_strokeq(-1)(oldq--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--
      (1.6,1.6)--(1.6,1.6)--(0,0));
  endgroup;
  expand_pbox;
enddef;
```

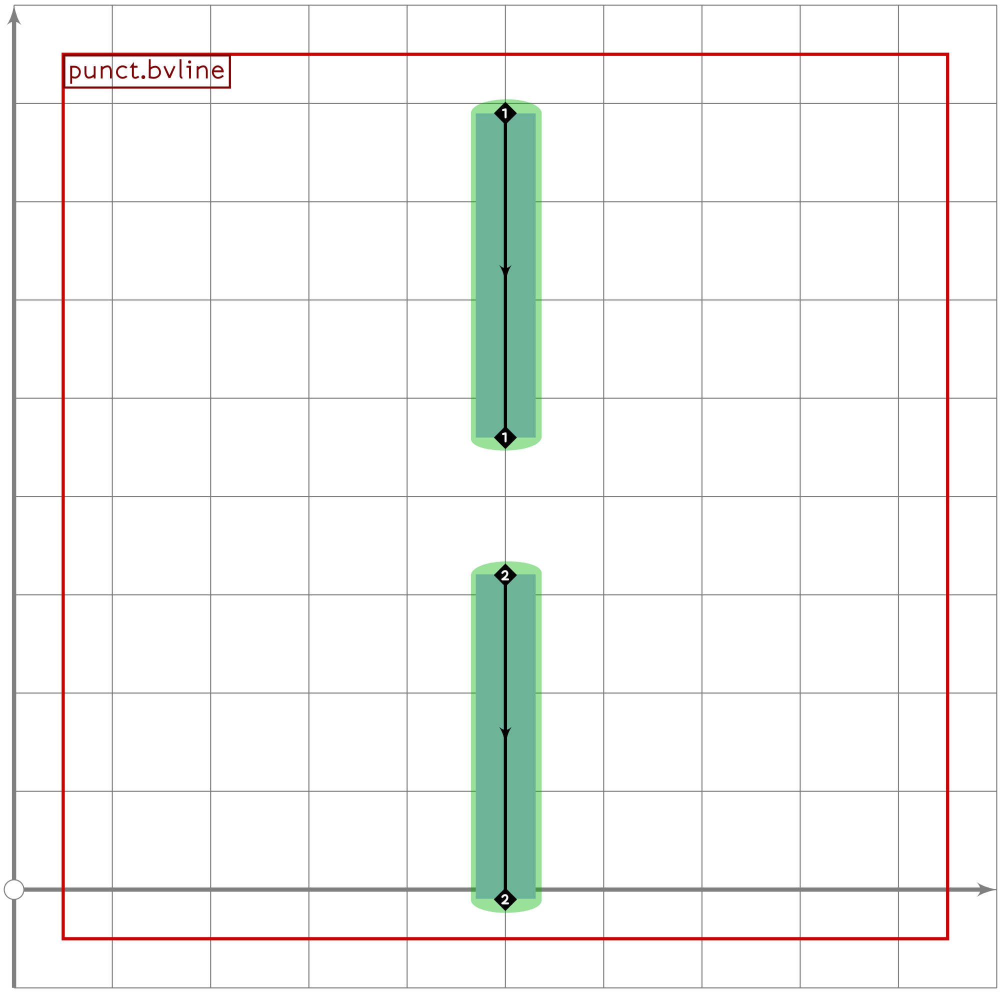


```

vardef punct.brace_left =
  push_pbox_toexpand("punct.brace_left");
  push_stroke((900,780){left}..
    (900-1.5*tsu_punct_size,680)..
    (900-1.1*tsu_punct_size,490)..
    (900-3.1*tsu_punct_size,390-0.25*tsu_punct_size){left}..
    (900-3.5*tsu_punct_size,390)..
    (900-3.1*tsu_punct_size,390+0.25*tsu_punct_size){right}..
    (900-1.1*tsu_punct_size,290)..
    (900-1.5*tsu_punct_size,100)..
    (900,0){right},
    (1.7,1.7)--(2,2)--(2,2)--
    (1.2,1.2)--(1.2,1.2)--(1.2,1.2)--
    (2,2)--(2,2)--(1.7,1.7));
  set_bosize(0,90);
  expand_pbox;
enddef;
```



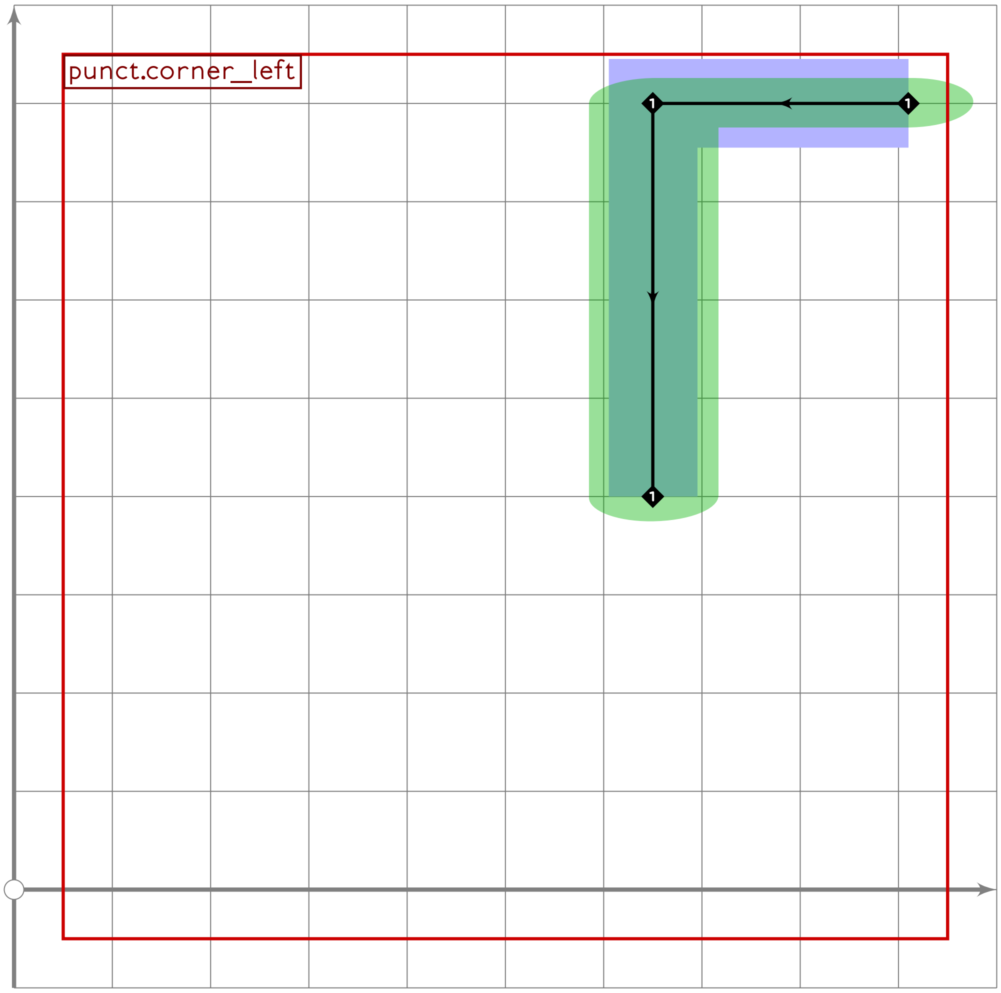
```
vardef punct.brace_right =
  push_pbox_toexpand(”punct.brace_right”);
  tsu_xform(identity rotatedaround (centre_pt,180))
    (punct.brace_left);
  expand_pbox;
enddef;
```

U+FFE4
tsuku.uniFFE4



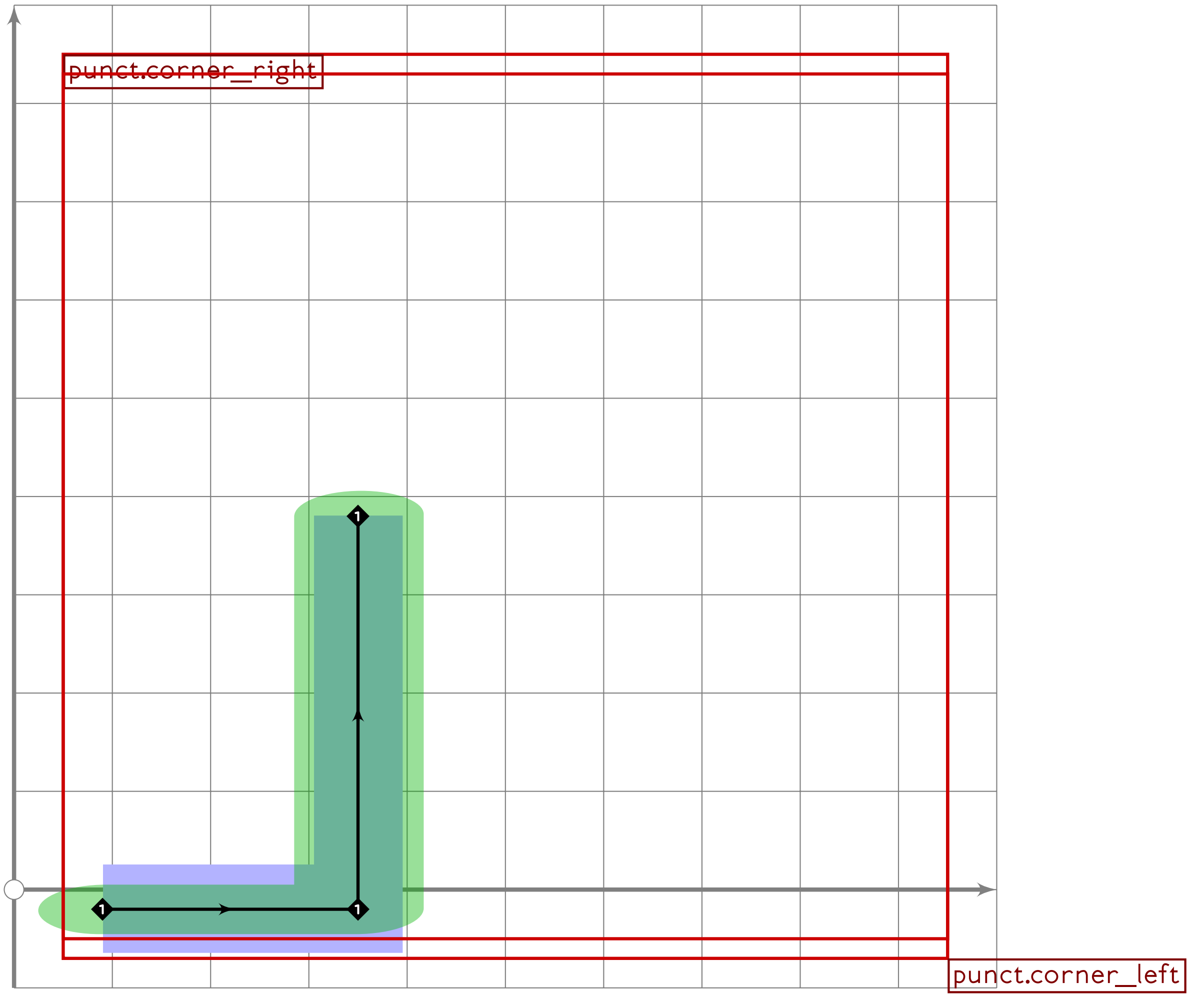
```

vardef punct.bvline =
  push_pbox_toexpand(”punct.bvline”);

  push_stroke((500,690+tsu_punct_size)–(500,390+0.7*tsu_punct_size),
    (1.6,1.6)–(1.6,1.6));
  set_bosize(0,90);

  push_stroke((500,390-0.7*tsu_punct_size)–(500,90-tsu_punct_size),
    (1.6,1.6)–(1.6,1.6));
  set_bosize(0,90);
  expand_pbox;
enddef;

vardef punct.make_comma(expr cpos,cang) =
  begingroup
    save x,y,t,u,xsp;
    numeric x[],y[];
    transform t,u;
    xsp:=sp;
    sp:=1;
    t:=tsu_rescale_xform;
    sp:=xsp;

    x1=0.8[x2,x4];
    (x2+x4)/2=x3=0;
    (x2-x4)=0.45*(y3-y1)=tsu_punct_size;
    x5=x3;

    y2=y4=0.32[y3,y1]=0;
    y5-y4=0.73*(y4-y3);

    push_stroke(z1{curl 0.2}..tension 1.2..z2..z3..z4{dir 280}..
        z5..{curl 0.2}(point 0.8 of (z1{curl 0.2}..tension 1.2..
        z2..z3..z4{dir 280})),
      (2,2)--(1,1)--(2,2)--(2,2)--(2,2)--(2,2)--(2,2));
    replace_strokep(0)((point 4.2 of oldp)--oldp);
    (0,0) transformed u=llcorner get_strokep(0);
    (1,0) transformed u=lrcorner get_strokep(0);
    (0,1) transformed u=ulcorner get_strokep(0);
    replace_strokep(0)(oldp rotated (cang-6) shifted (cpos transformed t)
      transformed inverse t);
    u:=u scaled 1.3 rotated (cang-6) shifted (cpos transformed t)
      transformed inverse t;
    set_botip(0,1,0);

    if tsu_pbrush_size>=30:
      replace_strokep(0)(subpath (0.03,5.75) of oldp);
      set_bosize(0,40);
      push_lcblob((subpath (1.6,4.9) of get_strokep(0))--cycle);
    else:
      replace_strokep(0)(subpath (0,5.2) of oldp);
      set_bosize(0,80);
    fi;
    push_pbox_explicit("punct.make_comma",u);
  endgroup;
enddef;

vardef punct.make_revcomma(expr cpos,cang) =
  begingroup
    save x,y,t,u,xsp;
    numeric x[],y[];
    transform t,u;
    xsp:=sp;
    sp:=1;
    t:=tsu_rescale_xform;
    sp:=xsp;

    x1=0.8[x2,x4];
    (x2+x4)/2=x3=0;
    (x2-x4)=0.45*(y3-y1)=tsu_punct_size;
    x5=x3;

    y2=y4=0.32[y3,y1]=0;
    y5-y4=0.73*(y4-y3);

    push_stroke(z1{curl 0.2}..tension 1.2..z2..z3..z4{dir 280}..
        z5..{curl 0.2}(point 0.8 of (z1{curl 0.2}..tension 1.2..
        z2..z3..z4{dir 280})),
      (2,2)--(1,1)--(2,2)--(2,2)--(2,2)--(2,2)--(2,2));
    replace_strokep(0)((point 4.2 of oldp)--oldp);
    (0,0) transformed u=llcorner get_strokep(0);
    (1,0) transformed u=lrcorner get_strokep(0);
    (0,1) transformed u=ulcorner get_strokep(0);
    replace_strokep(0)(reverse (oldp rotated -6 reflectedabout(up,down)));
    replace_strokep(0)(oldp rotated (cang-6) shifted (cpos transformed t)
      transformed inverse t);
    u:=u scaled 1.3 rotated (cang-6) shifted (cpos transformed t)
      transformed inverse t;
    set_botip(0,1,0);

    if tsu_pbrush_size>=30:
      replace_strokep(0)(subpath (0.25,5.97) of oldp);
      set_bosize(0,40);
      push_lcblob((subpath (1.1,4.4) of get_strokep(0))--cycle);
    else:
      replace_strokep(0)(subpath (0.8,6) of oldp);
      set_bosize(0,80);
    fi;
    push_pbox_explicit(”punct.make_revcomma”,u);
  endgroup;
enddef;
```

U+300C
tsuku.uni300C

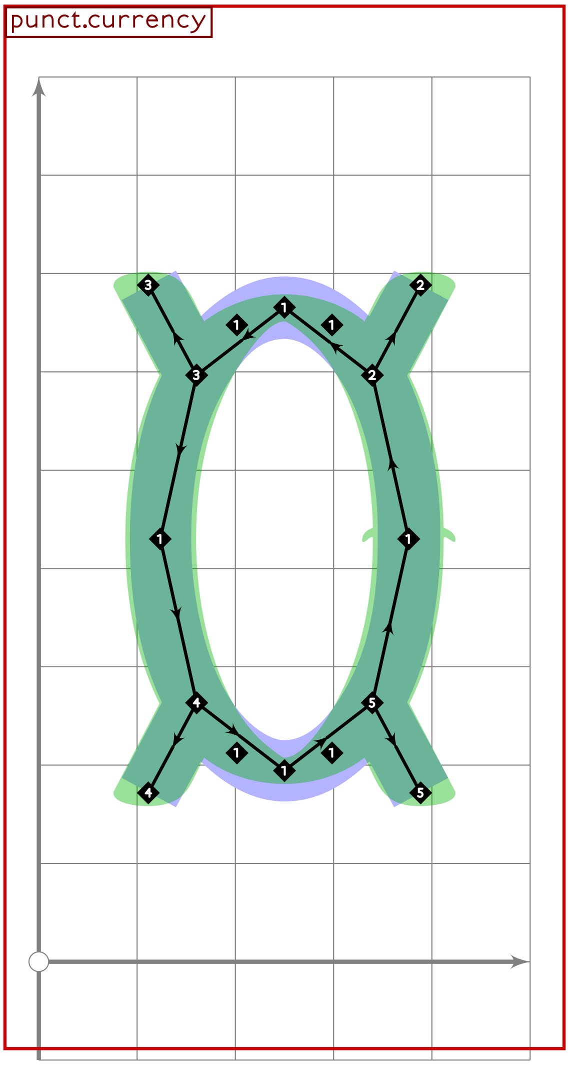


```

vardef  punct.corner_left =
  push_pbox_toexpand(”punct.corner_left”);
  push_stroke((910,800)--(650,800)--(650,400),(2,2)--(2,2)--(2,2));
  set_bosize(0,120);
  set_botip(0,1,1);
  expand_pbox;
enddef;
```

U+300D
tsuku.uni300D



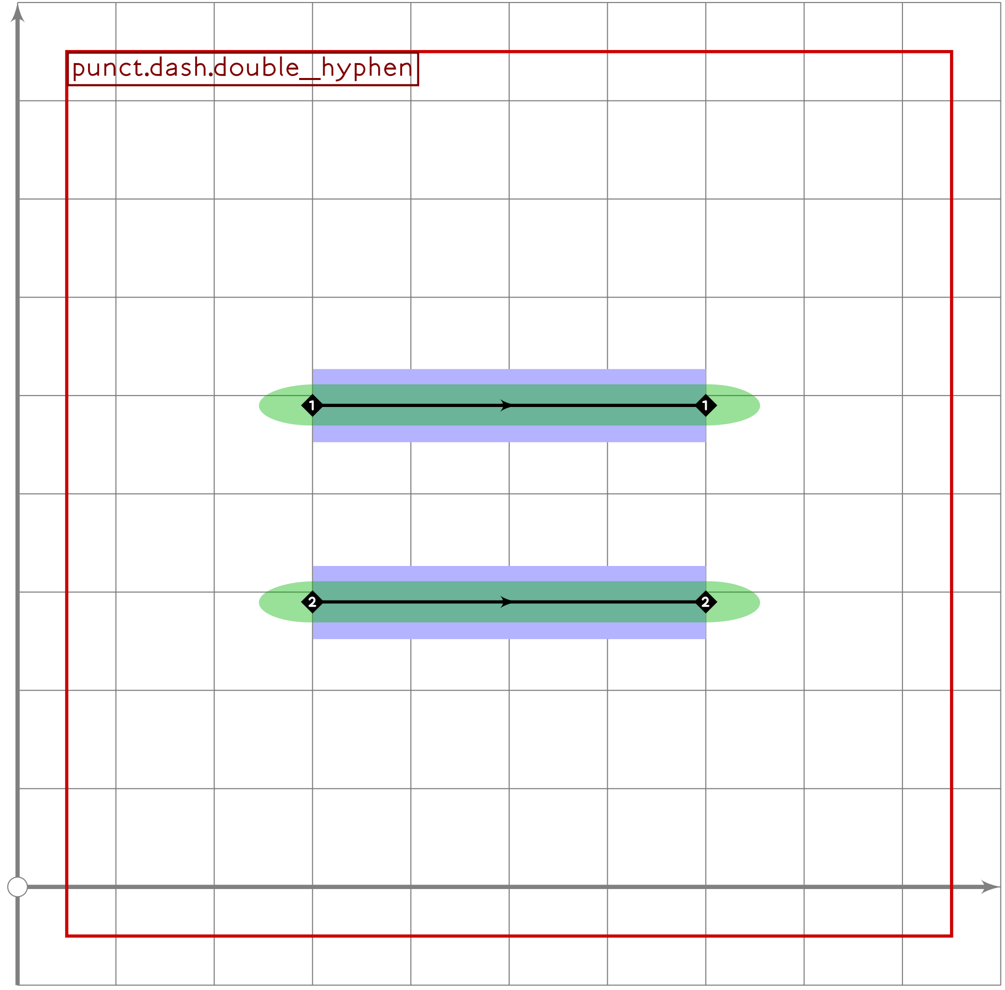
```
vardef punct.corner_right =
  push_pbox_toexpand("punct.corner_right");
  tsu_xform(identity rotatedaround (centre_pt,180))
    (punct.corner_left);
  expand_pbox;
enddef;
```



```
vardef punct.currency =
  push_pbox_toexpand(”punct.currency”);

  push_stroke(fullcircle scaled (4*tsu_punct_size) shifted centre_pt,
    (1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)–cycle);
  set_bosize(0,90);

  push_stroke(((1,0)–(1.55,0)) rotated 45
      scaled (2*tsu_punct_size) shifted centre_pt,
    (1.6,1.6)–(1.6,1.6));
  set_bosize(0,90);

  push_stroke(((1,0)–(1.55,0)) rotated 135
      scaled (2*tsu_punct_size) shifted centre_pt,
    (1.6,1.6)–(1.6,1.6));
  set_bosize(0,90);

  push_stroke(((1,0)–(1.55,0)) rotated 225
```

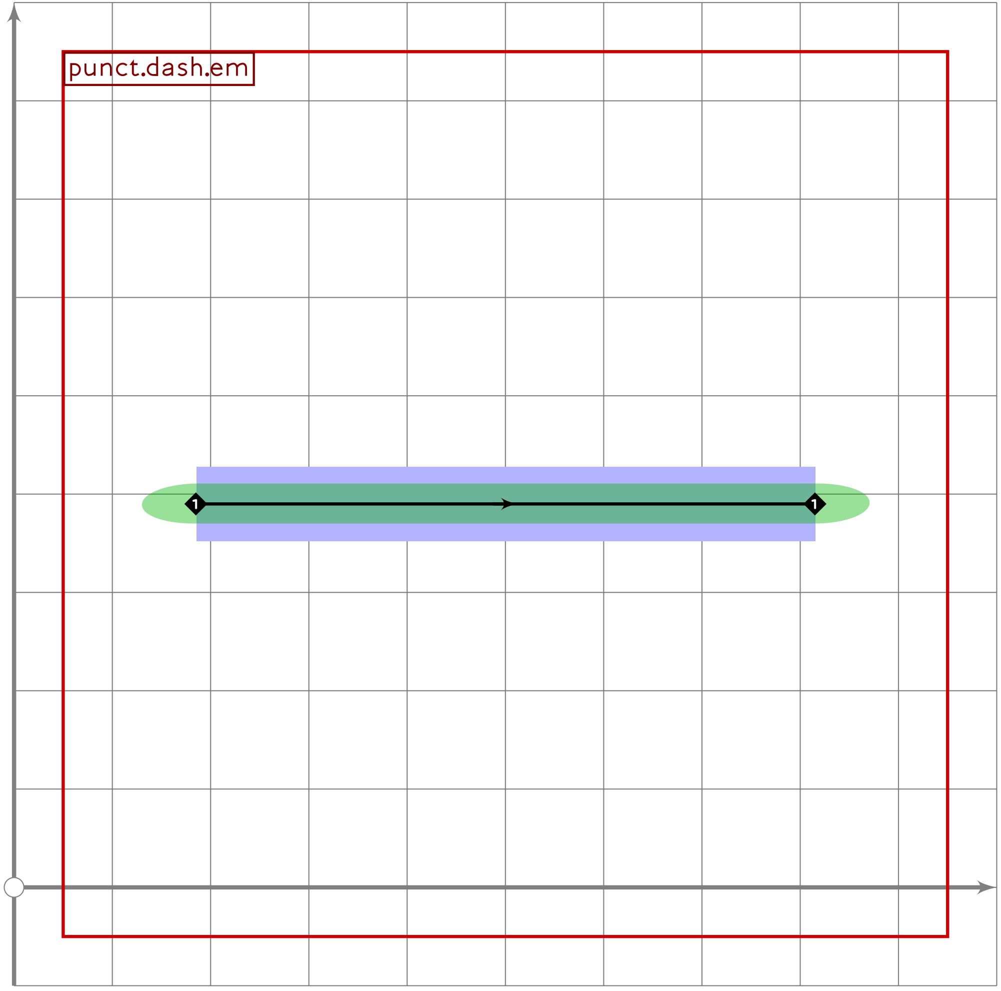
U+30A0
tsuku.uni30A0

```
        scaled (2*tsu_punct_size) shifted centre_pt,
      (1.6,1.6)--(1.6,1.6));
    set_bosize(0,90);

    push_stroke(((1,0)--(1.55,0)) rotated 315
        scaled (2*tsu_punct_size) shifted centre_pt,
      (1.6,1.6)--(1.6,1.6));
    set_bosize(0,90);
    expand_pbox;
enddef;
```



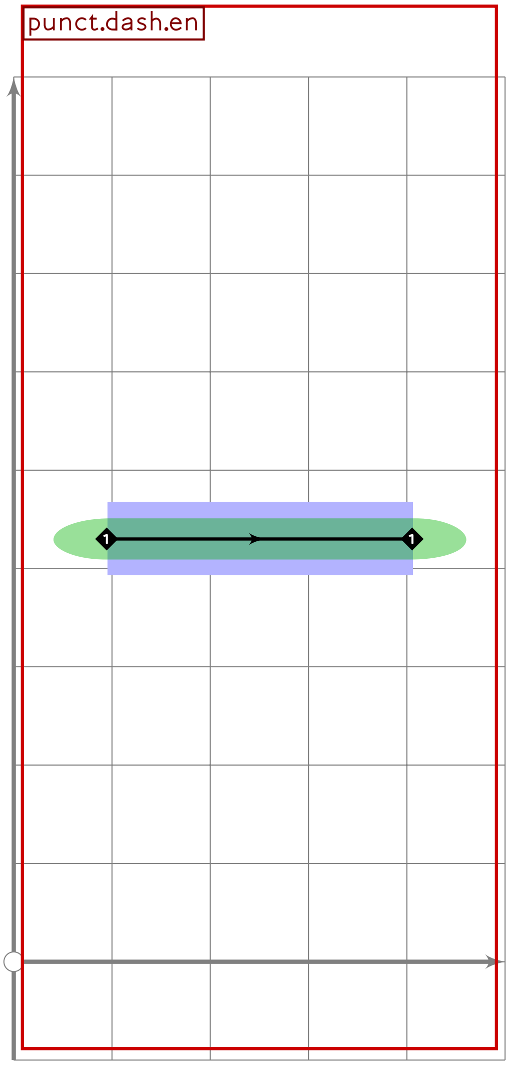
```

vardef punct.dash.double_hyphen =
    push_pbox_toexpand("punct.dash.double_hyphen");

    (z1+z4)/2=centre_pt;
    x2-x1=400;
    y1-y3=200;
    y1=y2;
    y3=y4;
    x1=x3;
    x2=x4;
```

```

  push_stroke(z1--z2,(2,2)--(2,2));
  push_stroke(z3--z4,(2,2)--(2,2));
  expand_pbox;
enddef;
```

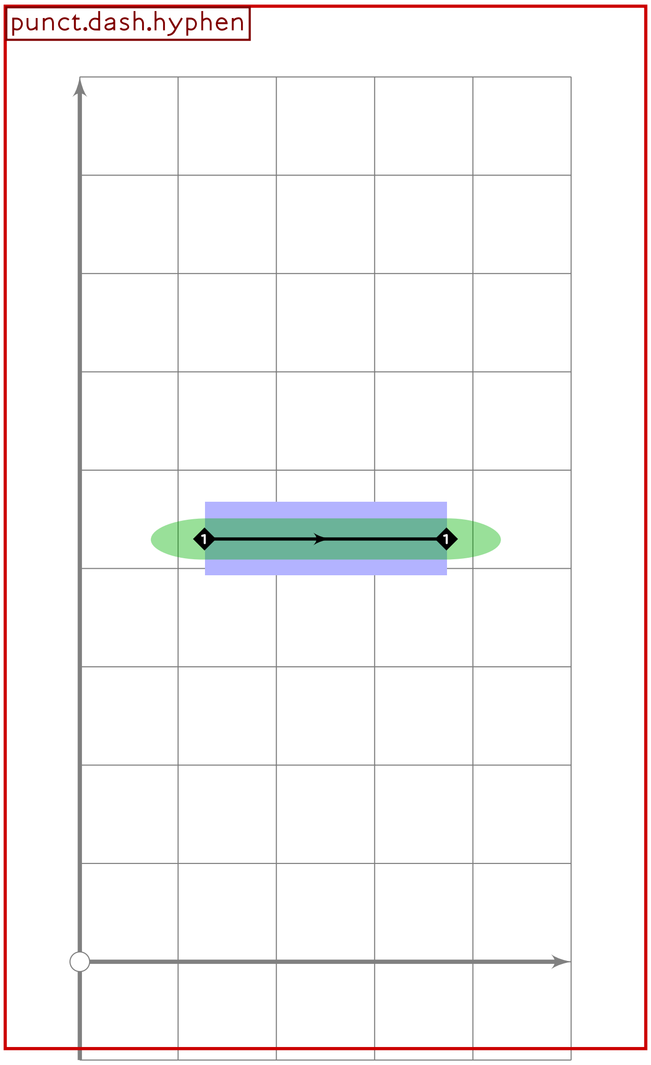


```

vardef punct.dash.em =
  push_pbox_toexpand(”punct.dash.em”);
  (z1+z2)/2=centre_pt;
  x2-x1=if is_proportional: 750 else: 630 fi;
  y1=y2;
  push_stroke(z1--z2,(2,2)--(2,2));
  expand_pbox;
enddef;
```

U+2012
tsuku.figuredash

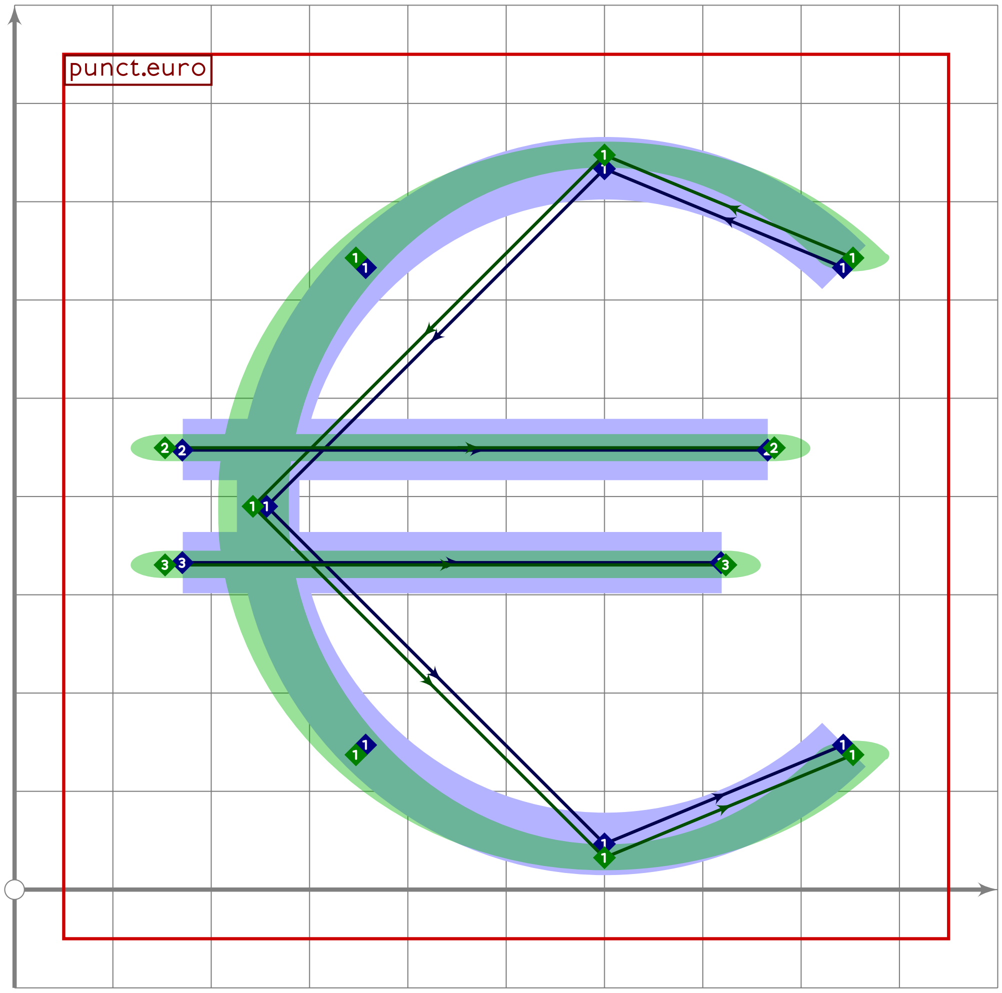


```

vardef  punct.dash.en  =
   push_pbox_toexpand(”punct.dash.en”);
   (z1+z2)/2=centre_pt;
   x2-x1=580;
   y1=y2;
   push_stroke(z1--z2,(2,2)--(2,2));
   expand_pbox;
enddef;
```



```
vardef punct.dash.hyphen =
  push_pbox_toexpand(”punct.dash.hyphen”);
  (z1+z2)/2=centre_pt;
  x2-x1=340;
  y1=y2;
  push_stroke(z1–z2,(2,2)–(2,2));
  expand_pbox;
enddef;

vardef punct.dash.long =
  push_pbox_toexpand(”punct.dash.long”);
  (z1+z2)/2=centre_pt;
  x2-x1=340;
  y1=y2;
  push_stroke(z1–z2,(2,2)–(2,2));
  expand_pbox;
enddef;
```

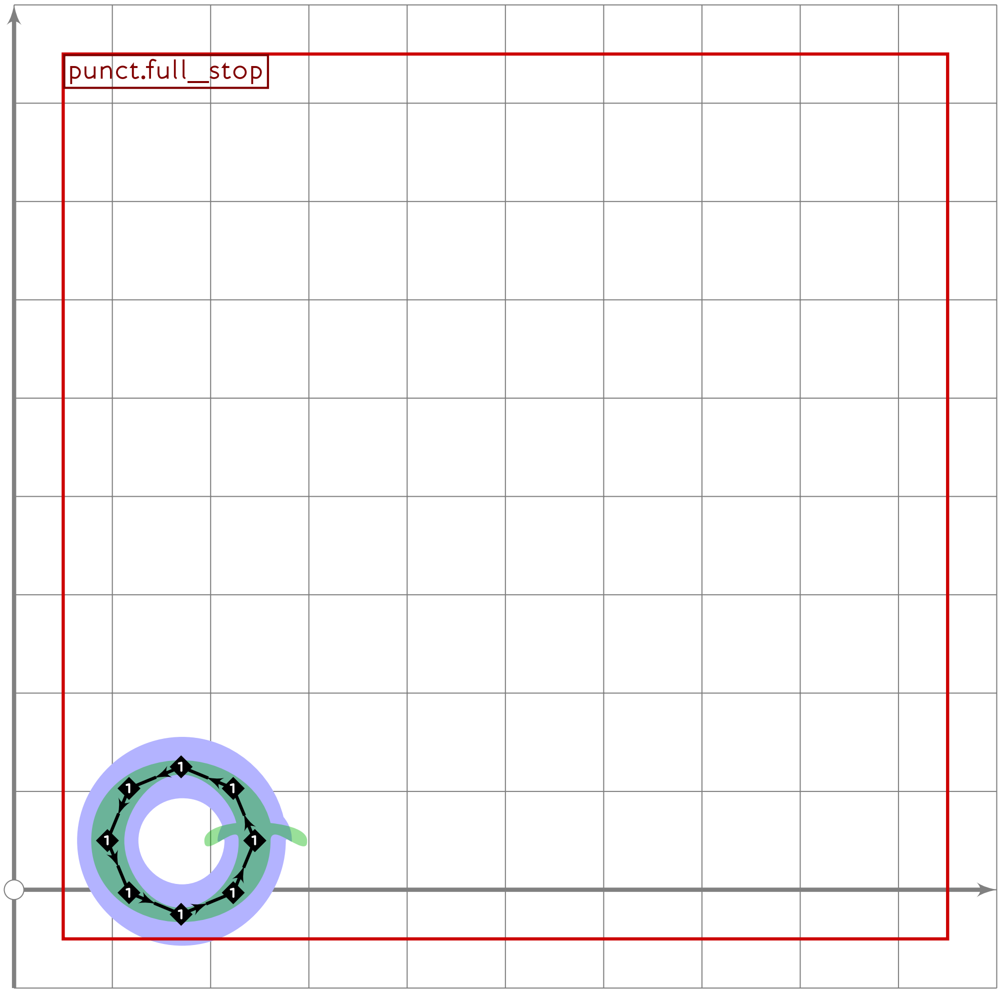
```
vardef punct.dividedby(expr t) =
  push_stroke(((-1,0)--(1,0)) transformed t,(2,2)--(2,2));
  set_bosize(0,90);

  push_lcblob(fullcircle scaled (0.65*tsu_punct_size/xxpart t)
    shifted (0,0.9) transformed t);
  push_lcblob(fullcircle scaled (0.65*tsu_punct_size/xxpart t)
    shifted (0,-0.9) transformed t);

  push_pbox_explicit("punct.dividedby",
    identity shifted (-0.5,-0.5) scaled 2.4 transformed t);
enddef;

vardef punct.equals(expr t) =
  push_stroke(((-1,0.667)--(1,0.667)) transformed t,(2,2)--(2,2));
  set_bosize(0,90);

  push_stroke(((-1,-0.667)--(1,-0.667)) transformed t,(2,2)--(2,2));
  set_bosize(0,90);

  push_pbox_explicit("punct.equals",
    identity shifted (-0.5,-0.5) xyscaled (2.4,1.8) transformed t);
enddef;
```



```
vardef punct.euro =
  push_pbox_toexpand(”punct.euro”);

  push_stroke((subpath (0.5,3.5) of ((1,0)..(0,1)..(-1,0)..(0,-1)..cycle))
      scaled ((latin_wide_high_r-latin_wide_low_r)/2)
      shifted (centre_pt+(100,0)),
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  set_bosize(0,90);

  push_stroke(((-1.25,0.1667)--
      (((0,0.1667)--(1,0.1667)) intersectionpoint ((0.707,0.707)--(0,-1))))
      scaled ((latin_wide_high_r-latin_wide_low_r)/2)
      shifted (centre_pt+(100,0)),
    (1.6,1.6)--(1.6,1.6));
  set_bosize(0,90);

  push_stroke(((-1.25,-0.1667)--
      (((0,-0.1667)--(1,-0.1667)) intersectionpoint ((0.707,0.707)--(0,-1))))
      scaled ((latin_wide_high_r-latin_wide_low_r)/2)
      shifted (centre_pt+(100,0)),
```

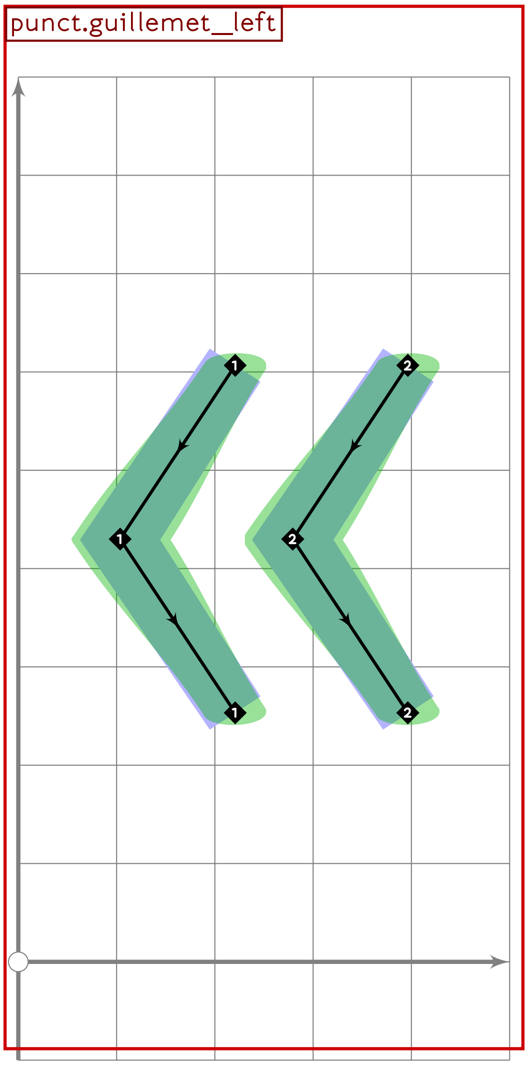
U+3002
tsuku.uni3002

```
    (1.6,1.6)--(1.6,1.6));
  set_bosize(0,90);
  expand_pbox;
enddef;

```



```
% this is *ideographic* full stop, not Latin period
vardef punct.full_stop =
  push_pbox_toexpand(”punct.full_stop”);

  if tsu_pbrush_size>=tsu_punct_size:
    push_lcblob(fullcircle
      xscaled (1.5*tsu_punct_size+tsu_pbrush_size)
      yscaled (1.5*tsu_punct_size+tsu_pbrush_size*tsu_pbrush_shape)
      rotated tsu_pbrush_angle
      shifted (170,50));
  else:
    push_stroke(fullcircle scaled (1.5*tsu_punct_size) shifted (170,50),
      (1,1)--(1,1)--(1,1)--(1,1)--cycle);
  fi;
  expand_pbox;
enddef;
```

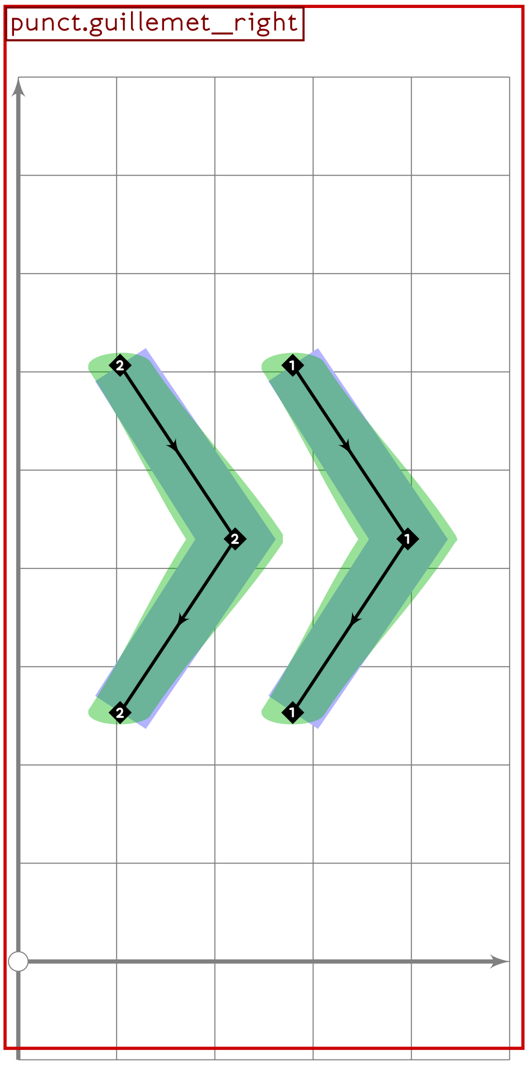
```

vardef punct.greater_than(expr t) =
  push_stroke(((-1,1)--(1,0)--(-1,-1)) transformed t,(2,2)--(2,2)--(2,2));
  set_bosize(0,90);
  set_botip(0,1,1);

  push_pbox_explicit("punct.greater_than",
    identity shifted (-0.5,-0.5) scaled 2.4 transformed t);
enddef;
```



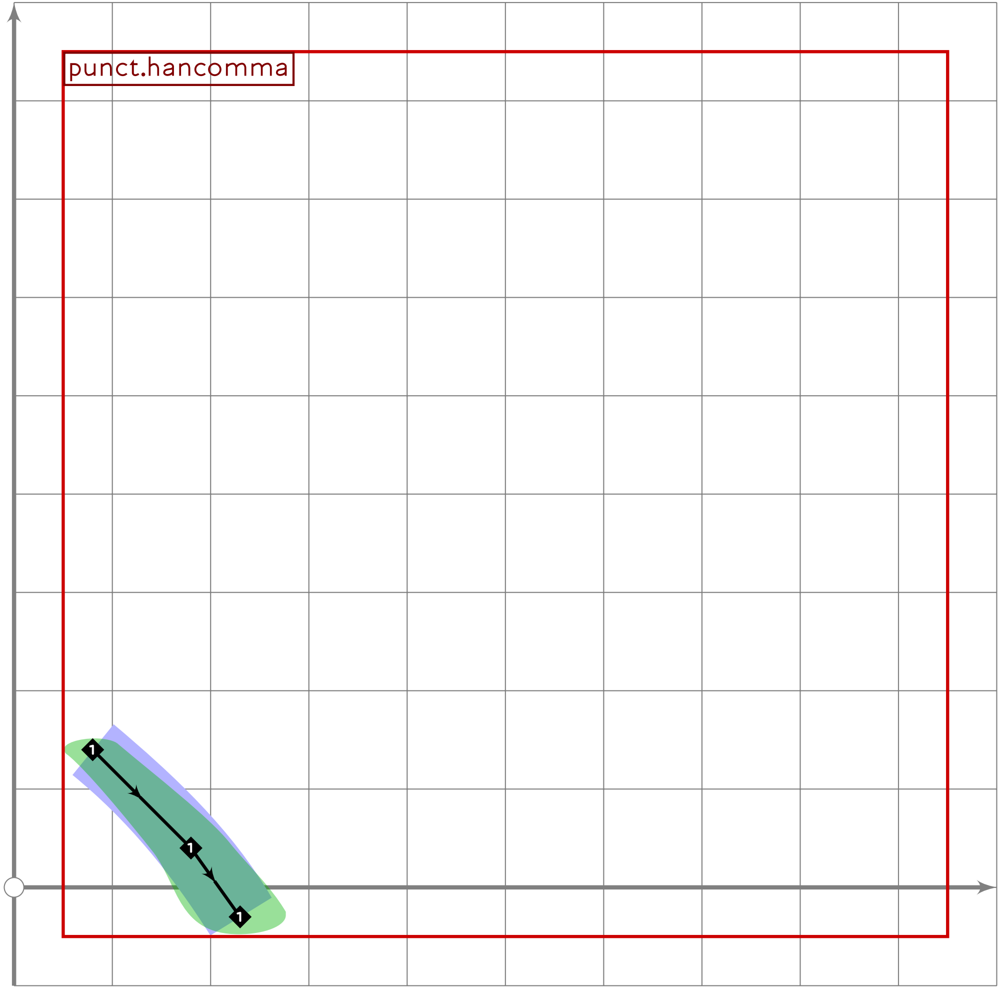
```

vardef punct.guillemet_left =
  push_pbox_toexpand("punct.guillemet_left");

  push_stroke(((-0.5,1.5)--(-2.5,0)--(-0.5,-1.5))
      scaled tsu_punct_size shifted centre_pt,
    (1.5,1.5)--(2,2)--(1.5,1.5));
  set_bosize(0,90);
  set_botip(0,1,1);

```

U+00BB
tsuku.guillemotright

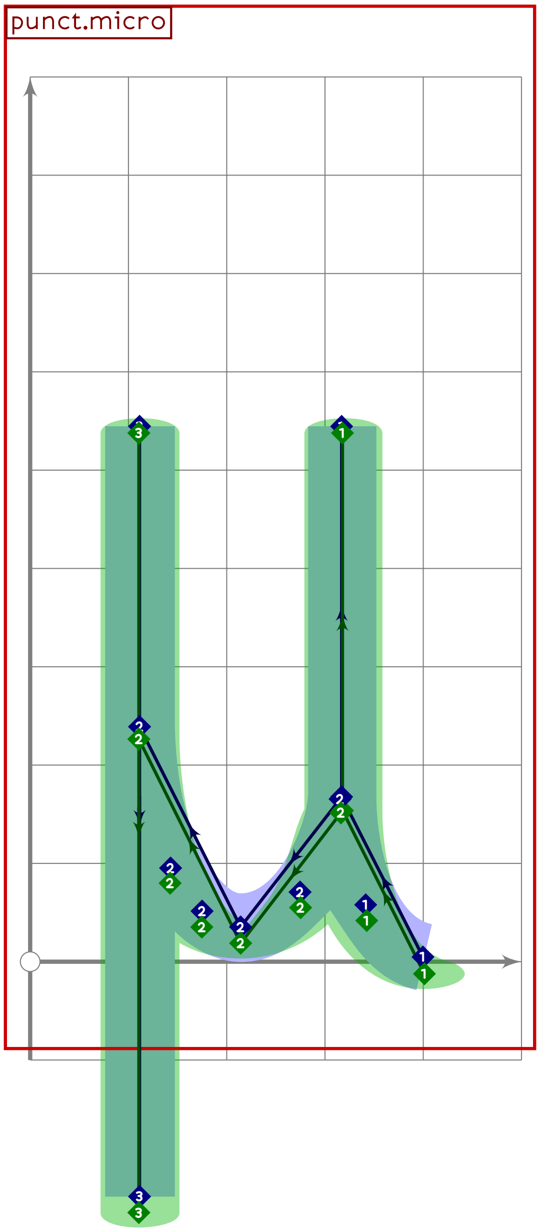
```
  push_stroke(((2.5,1.5)--(0.5,0)--(2.5,-1.5))
      scaled tsu_punct_size shifted centre_pt,
    (1.5,1.5)--(2,2)--(1.5,1.5));
  set_bosize(0,90);
  set_botip(0,1,1);
  expand_pbox;
enddef;
```



```

vardef punct.guillemet_right =
  push_pbox_toexpand("punct.guillemet_right");

  push_stroke(((0.5,1.5)--(2.5,0)--(0.5,-1.5))
      scaled tsu_punct_size shifted centre_pt,
    (1.5,1.5)--(2,2)--(1.5,1.5));
  set_bosize(0,90);
  set_botip(0,1,1);

  push_stroke(((-2.5,1.5)--(-0.5,0)--(-2.5,-1.5))
      scaled tsu_punct_size shifted centre_pt,
    (1.5,1.5)--(2,2)--(1.5,1.5));
  set_bosize(0,90);
  set_botip(0,1,1);
  expand_pbox;
enddef;
```

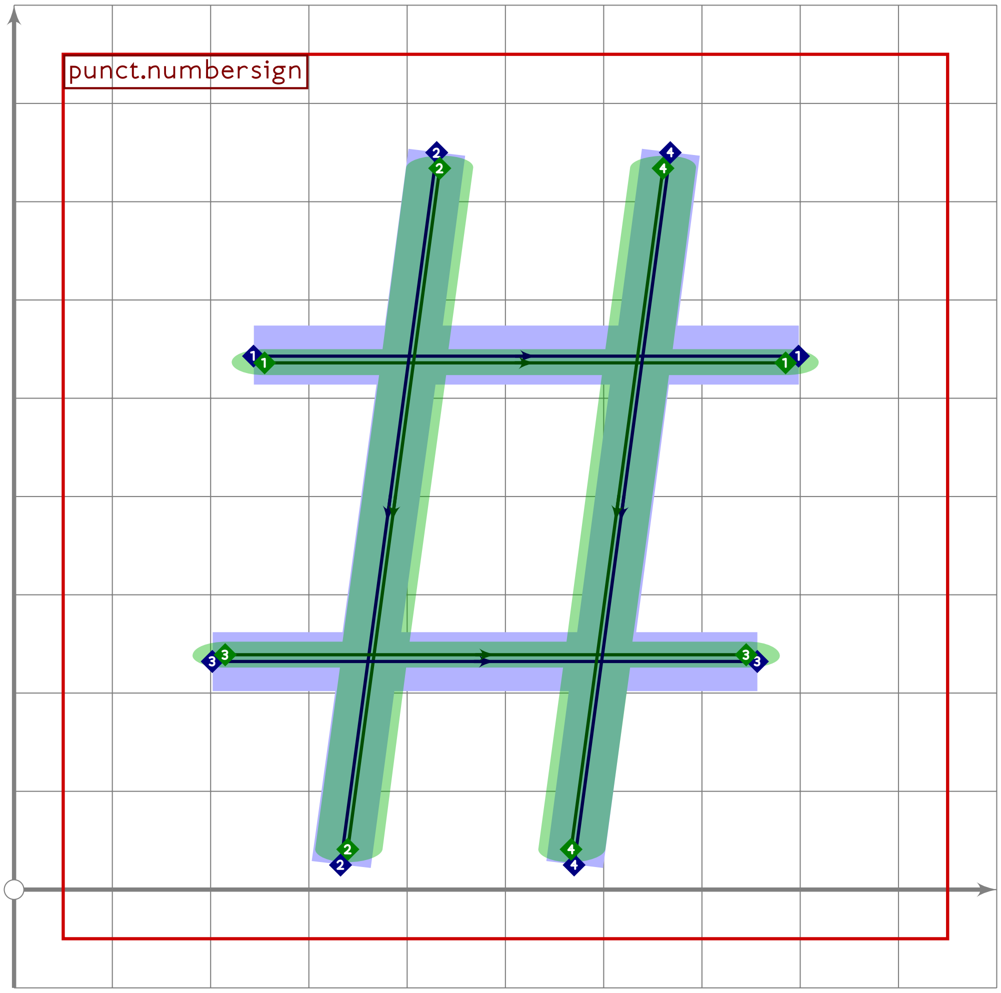


```

vardef punct.hancomma =
  push_pbox_toexpand("punct.hancomma");
  push_stroke((80,140)..(180,40)..(230,-30),(1.3,1.3)..(1.6,1.6)..(1.8,1.8));
  expand_pbox;
enddef;

vardef punct.hminus(expr t) =
  push_stroke(((-1,0)--(1,0)) transformed t,(2,2)--(2,2));

  push_pbox_explicit("punct.hminus",
    identity shifted (-0.5,-0.5) xyscaled (2.4,0.6) transformed t);
enddef;

vardef punct.less_than(expr t) =
  push_stroke((((1,1)--(-1,0)--(1,-1)) transformed t,(2,2)--(2,2)--(2,2));
  set_bosize(0,90);
  set_botip(0,1,1);

  push_pbox_explicit("punct.less_than",
    identity shifted (-0.5,-0.5) scaled 2.4 transformed t);
enddef;

```

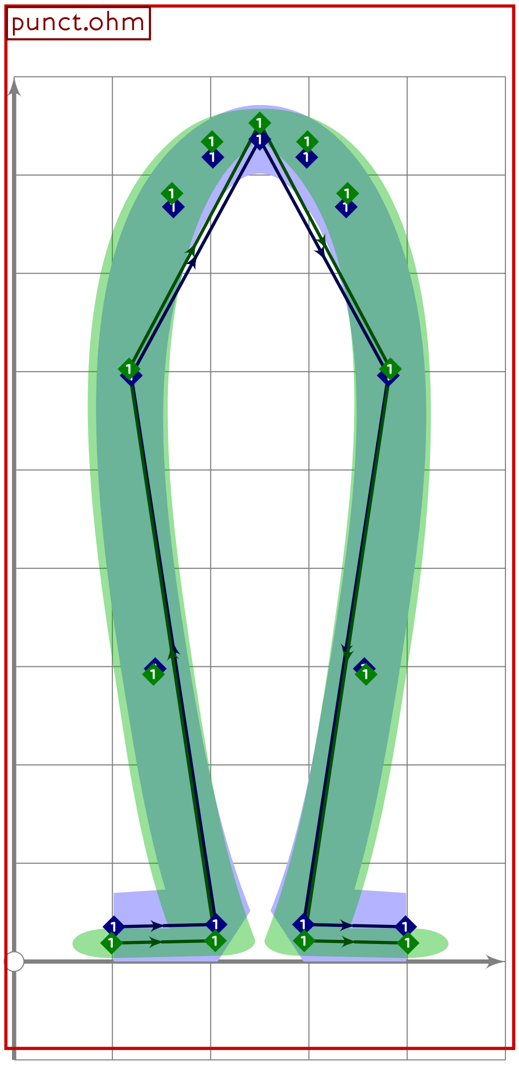


```
% in the future, this will probably become greek.lowmu
vardef punct.micro =
  push_pbox_toexpand("punct.micro");

  x1-x2=y2-y1;
  (x2+x6)/2=450;
  (x2-x6)=(y3-y1)*0.75;
  x3=x2=x4;
  x5=0.5[x4,x6];
  x7=x8=x6;

  y1=(-0.06)[y5,y3];
  y2=0.26[y5,y3];
  y3=y7=latin_wide_xheight_v;
  y4=0.73[y3,y5];
  y5=latin_wide_low_h;
  y6=0.60[y3,y5];
  y8=latin_wide_desc_v;

  push_stroke(z1{dir 173}..{up}z2--z3,(1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  push_stroke(subpath (0.03,2) of (z4..z5{left}..z6{dir 93}),
    (1.6,1.6)--(1.6,1.6)--(1.6,1.6));
  push_stroke(z7--z8,(1.6,1.6)--(1.6,1.6));
  expand_pbox;
enddef;

vardef punct.notsign(expr t) =
  push_stroke(((-1,0)--(1,0)--(1,-1)) transformed t,(2,2)--(2,2)--(2,2));
  set_bosize(0,90);
  set_botip(0,1,1);

  push_pbox_explicit("punct.notsign",
    identity shifted (-0.5,-0.5) scaled 2.4 transformed t);
enddef;
```

U+FF03
tsuku.uniFF03



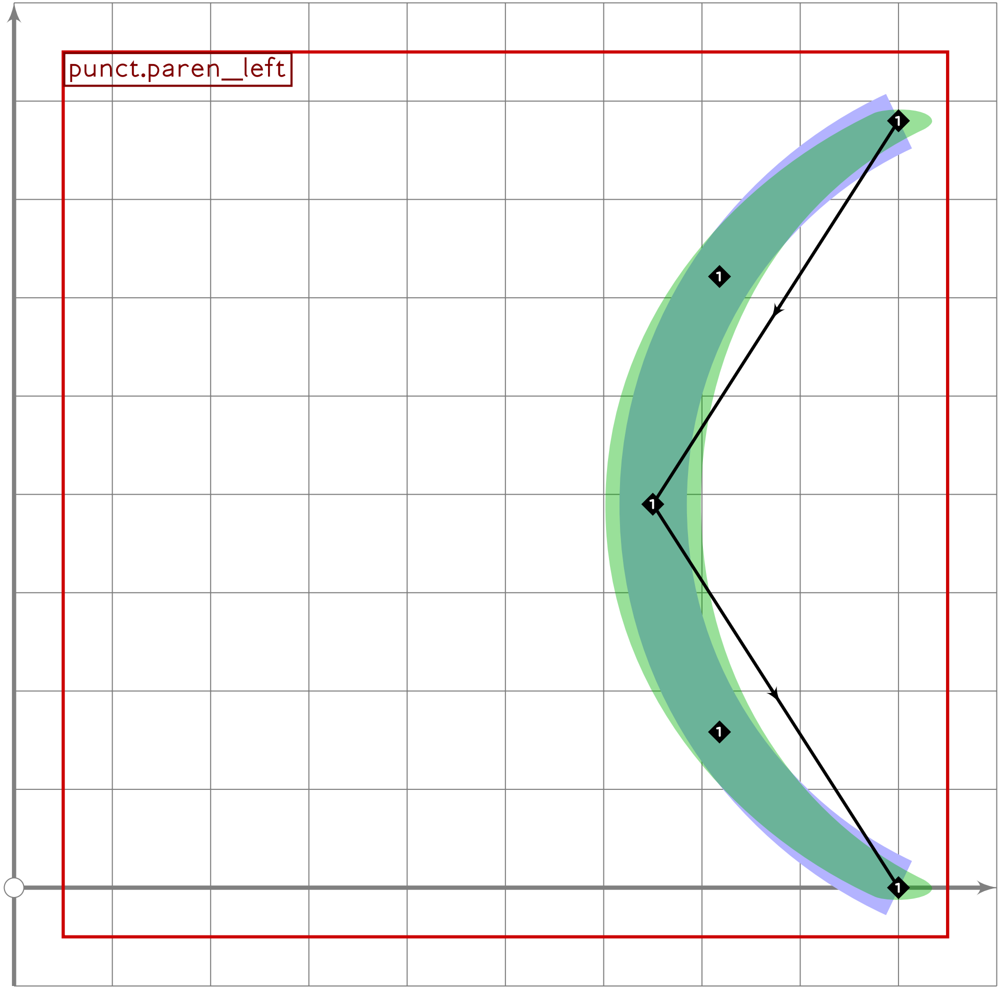
```
vardef punct.numbersign =
  push_pbox_toexpand(”punct.numbersign”);

  (x1+x2)/2=500;
  (x2-x1)=0.9*(y2-y1);
  x3=0.15[x1,x2];

  y1=latin_wide_low_v;
  y2=y3=latin_wide_high_v;

  transform xf_num;
  (0,0) transformed xf_num = z1;
  (3.5,3.5) transformed xf_num = z2;
  (0,3.5) transformed xf_num = z3;

  push_stroke(((0,2.5)–(3.5,2.5)) transformed xf_num,(1.6,1.6)–(1.6,1.6));
  set_bosize(0,85);
  push_stroke(((1,3.5)–(1,0)) transformed xf_num,(1.6,1.6)–(1.6,1.6));
  set_bosize(0,85);
  push_stroke(((0,1)–(3.5,1)) transformed xf_num,(1.6,1.6)–(1.6,1.6));
  set_bosize(0,85);
  push_stroke(((2.5,3.5)--(2.5,0)) transformed xf_num,(1.6,1.6)--(1.6,1.6));
  set_bosize(0,85);
  expand_pbox;
enddef;

```



```
% in the future, this will probably become greek.upomega
vardef punct.ohm =
  push_pbox_toexpand("punct.ohm");

  (x5+x3)/2=(x6+x2)/2=(x7+x1)/2=x4=500;
  x2=0.7[x1,x4];
  x7-x1=0.76*(y4-y1);
  x5-x3=0.67*(y4-y1);

  y1=y7=latin_wide_low_h;
  y2=y6=y1+2;
  y3=y5=0.7[y1,y4];
  y4=latin_wide_high_r;
```

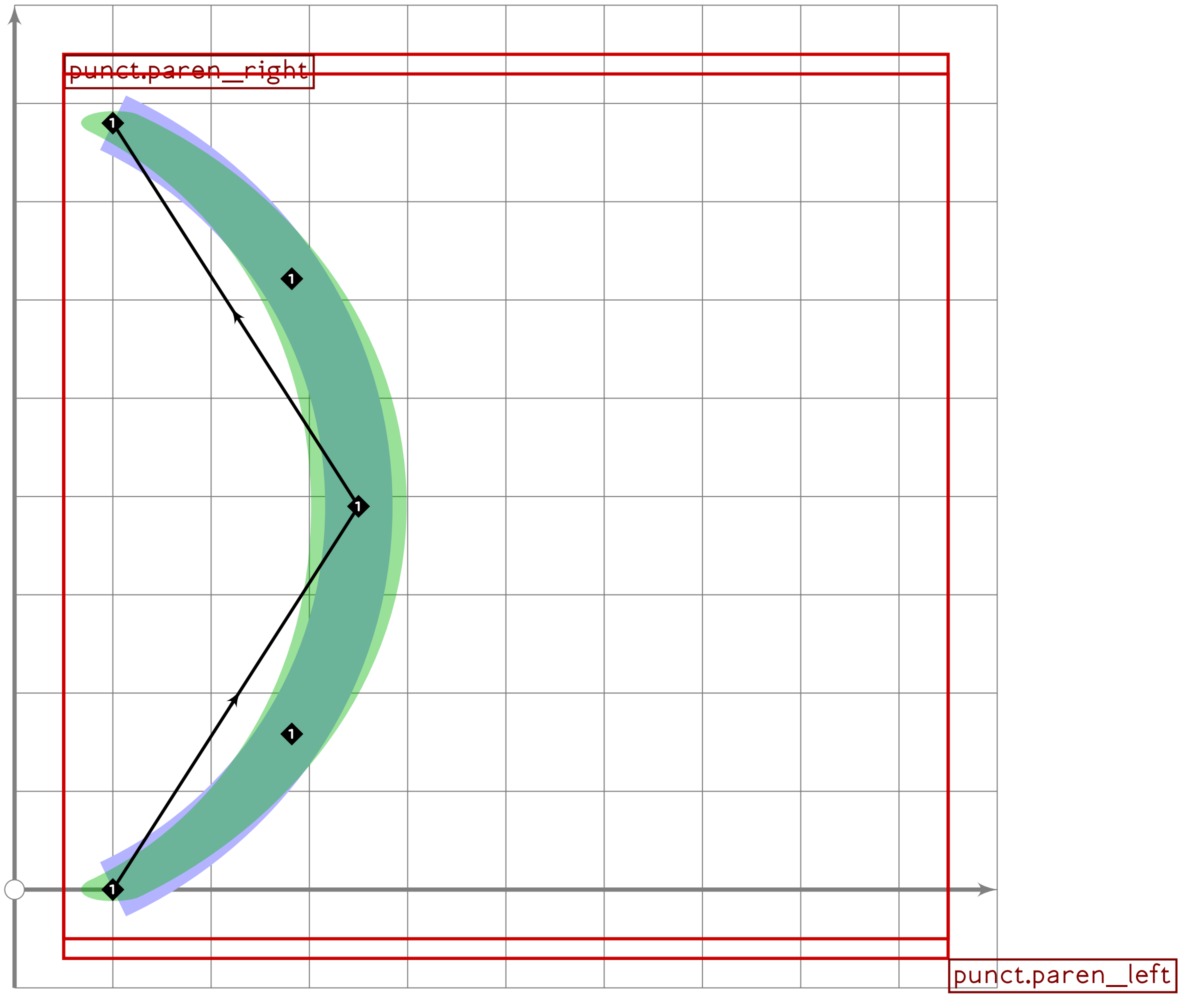
U+FF08
tsuku.uniFF08

```

  push_stroke(z1–z2..tension 1.5 and 1.3..z3..z4..
      z5..tension 1.3 and 1.5..z6–z7,
    (1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)–(1.6,1.6)–
      (1.6,1.6)–(1.6,1.6));
  set_botip(0,1,0);
  set_botip(0,5,0);
  expand_pbox;
enddef;
```



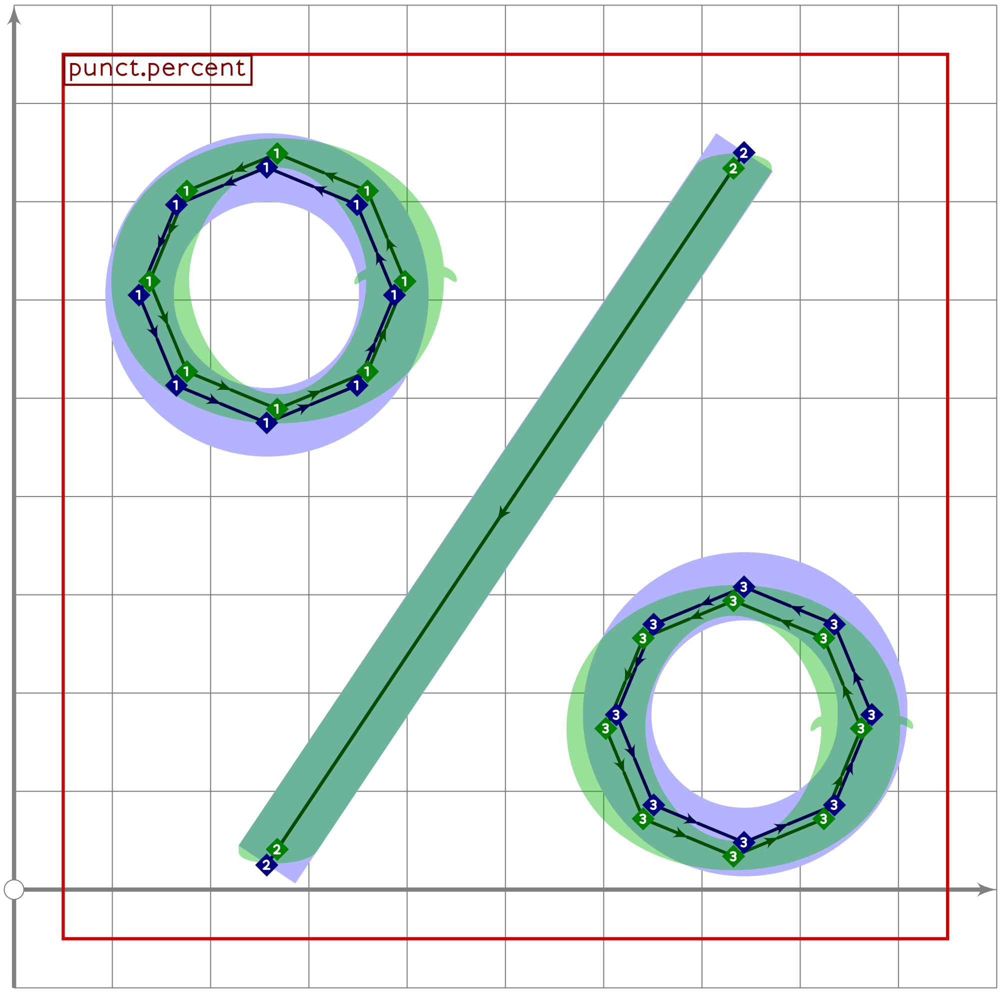
```

vardef punct.paren_left =
  push_pbox_toexpand(”punct.paren_left”);
  push_stroke((900,780)..(900-2.5*tsu_punct_size,390)..(900,0),
    (1.5,1.5)–(2,2)–(1.5,1.5));
  set_bosize(0,90);
  expand_pbox;
enddef;
```

U+FF09
tsuku.uniFF09

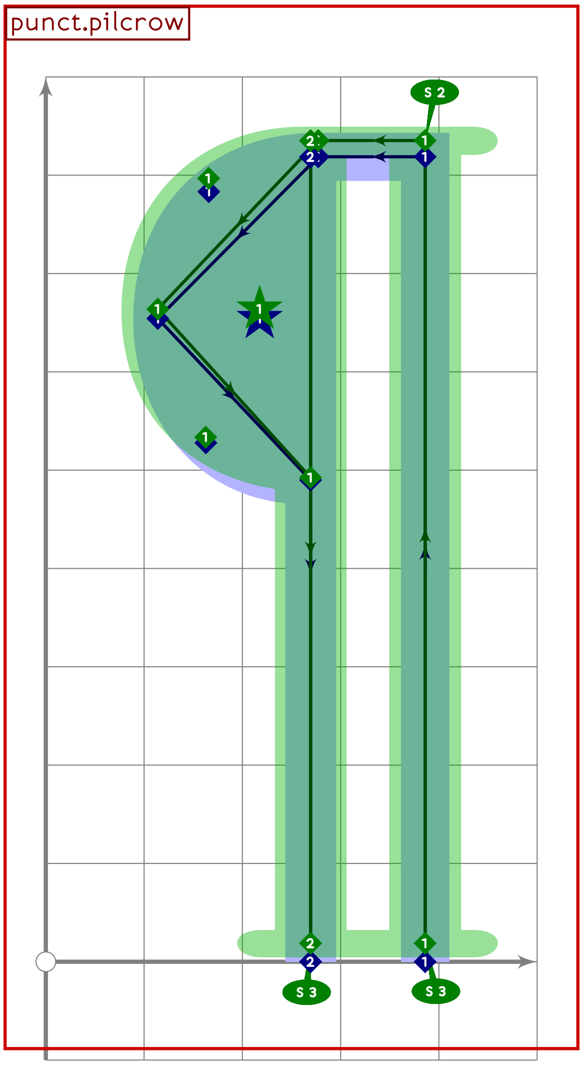


```
vardef punct.paren_right =
  push_pbox_toexpand("punct.paren_right");
  tsu_xform(identity rotatedaround (centre_pt,180))
    (punct.paren_left);
  expand_pbox;
enddef;
```

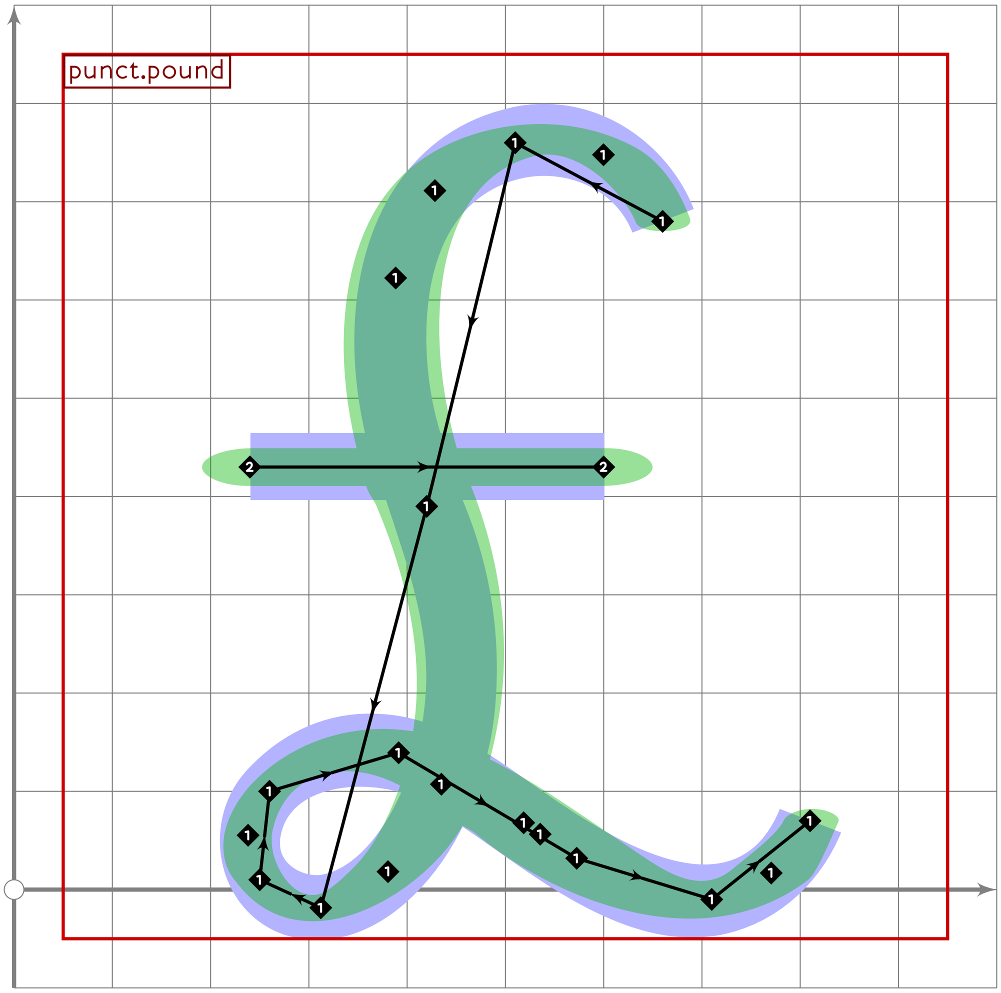
U+FF05
tsuku.uniFF05



```
vardef punct.percent =
  push_pbox_toexpand(”punct.percent”);

  (x1+x2)/2=500;
  (x1-x2)=0.67(y1-y2);

  y1=latin_wide_high_v;
  y2=latin_wide_low_v;

  push_stroke(fullcircle scaled 260 shifted (x2,latin_wide_high_r-130),
    (1.6,1.6)−(1.6,1.6)−(1.6,1.6)−(1.6,1.6)−cycle);

  push_stroke((z1−z2)
      shifted -centre_pt scaled (tsu_punct_size/100) shifted centre_pt,
    (1.6,1.6)−(1.6,1.6));

  push_stroke(fullcircle scaled 260 shifted (x1,latin_wide_low_r+130),
    (1.6,1.6)−(1.6,1.6)−(1.6,1.6)−(1.6,1.6)−cycle);
  expand_pbox;
enddef;
```

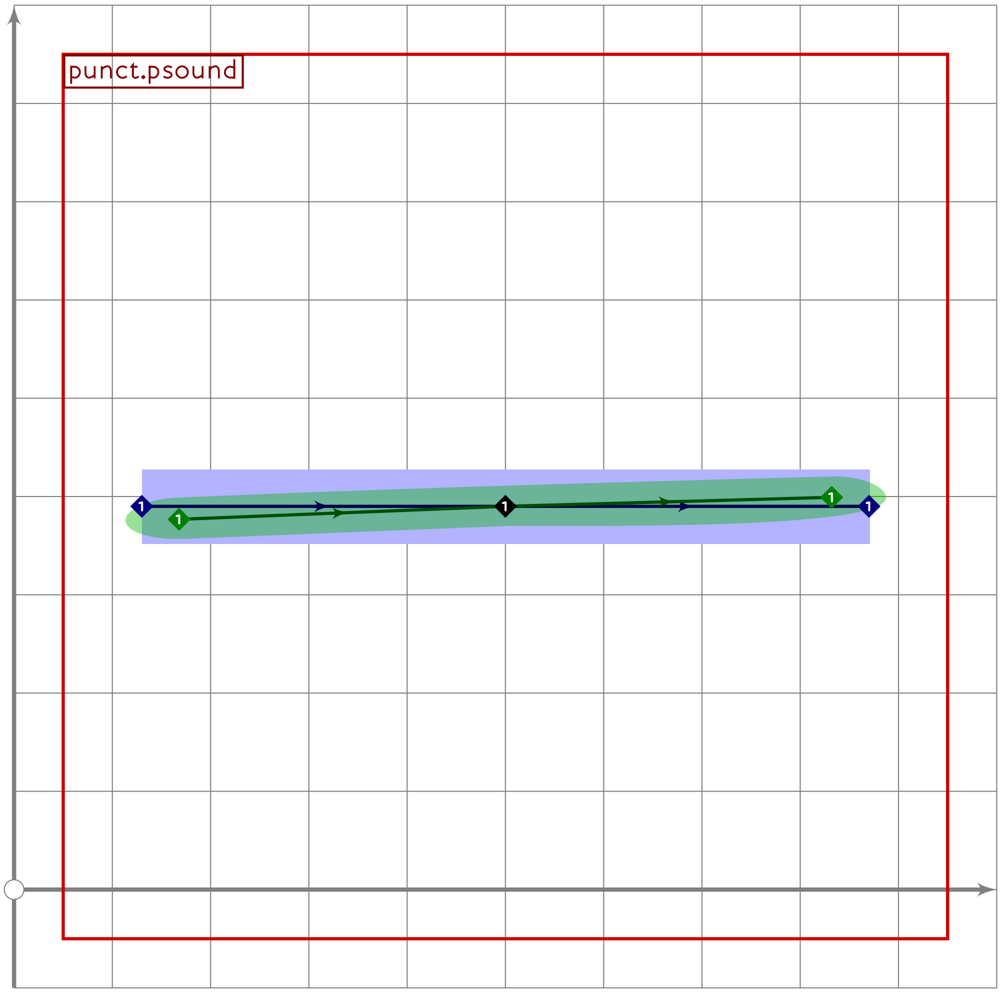


```
vardef punct.pilcrow =
  push_pbox_toexpand(”punct.pilcrow”);

  x1=x2=x6=710;
  x3=x5=x1-420*0.4;
  x4=x1-420;

  y1=latin_wide_low_v;
  y2=y3=latin_wide_high_h;
  y4=(y3+y5)/2;
  y5=y6=vmetric(0.58);

  x7=x8=x9=x1-1.8*tsu_punct_size;
  y7=y2+50;
  y8=y4;
  y9=y1;

  push_stroke(z1--z2--z3{left}..z4..{right}z5--z6,
    (2,2)--(2,2)--(2,2)--(2,2)--(2,2)--(2,2));
  replace_strokeq(0)(subpath (0,xpart (get_strokep(0) intersectiontimes
    (z8--z9))) of oldq);
  replace_strokep(0)(subpath (0,xpart (oldp intersectiontimes
    (z8--z9))) of oldp);
  set_bosize(0,67);
  set_botip(0,1,1);
  set_boserif(0,0,3);
  set_boserif(0,1,2);

  push_stroke(((z7--z8) intersectionpoint get_strokep(0))--z9,(2,2)--(2,2));
  set_bosize(0,67);
  set_boserif(0,1,3);

  if tsu_pbrush_size>=30:
    push_lcblob((subpath (xpart (get_strokep(-1) intersectiontimes (z7--z8)),
      infinity) of get_strokep(-1))--cycle);
  fi;
  expand_pbox;
enddef;

vardef punct.plus(expr t) =
  push_stroke(((-1,0)--(1,0)) transformed t,(2,2)--(2,2));
  push_stroke(((0,1)--(0,-1)) transformed t,(2,2)--(2,2));
  push_pbox_explicit("punct.plus",
    identity shifted (-0.5,-0.5) scaled 2.4 transformed t);
enddef;

vardef punct.plusminus(expr t) =
  push_stroke(((-1,0.25)--(1,0.25)) transformed t,(2,2)--(2,2));
  push_stroke(((0,1.25)--(0,-0.75)) transformed t,(2,2)--(2,2));
  push_stroke(((-1,-1.25)--(1,-1.25)) transformed t,(2,2)--(2,2));

  push_pbox_explicit("punct.plusminus",
    identity shifted (-0.5,-0.5) xyscaled (2.4,3.2) transformed t);
enddef;
```



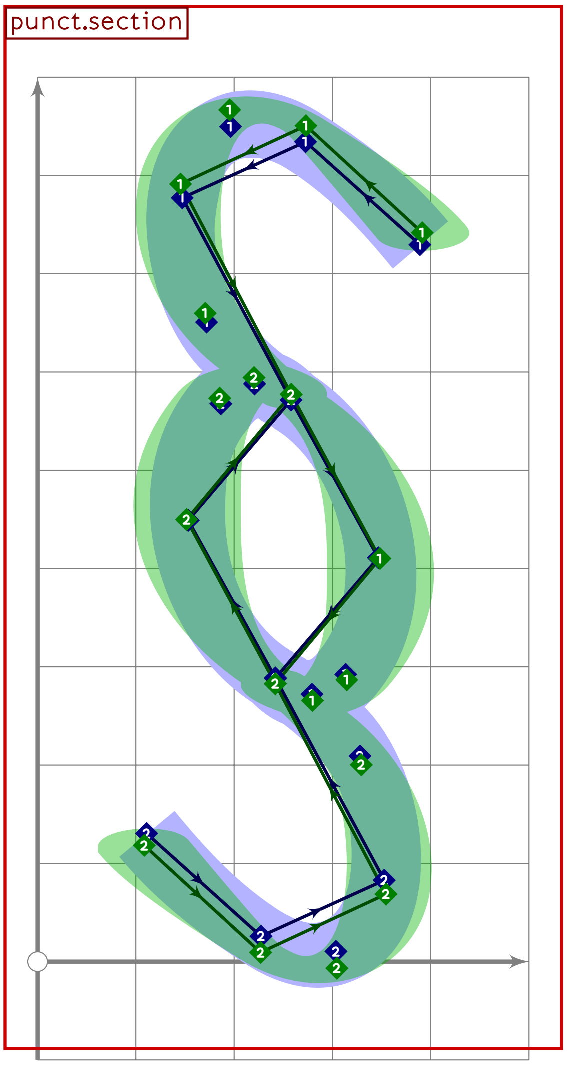
```
vardef  punct.pound =
  push_pbox_toexpand("punct.pound");

  push_stroke((660,680)..(510,760)..(420,390)..tension  1.1..(250,10)..
      (260,100)..(710,-10)..(810,70),
    (1.3,1.3)--(1.7,1.7)--(1.9,1.9)--(1.4,1.4)--(1.2,1.2)--
      (1.1,1.2)--(2,2)--(2.1,2.1)--(2,2)--(1.3,1.3));
  replace_strokep(0)(insert_nodes(oldp)(2.8,4.3,4.7));

  push_stroke((240,430)--(600,430),(2,2)--(2,2));
  set_bosize(0,90);
  expand_pbox;
enddef;
```

U+30FC
tsuku.uni30FC



```

vardef punct.psound =
  push_pbox_toexpand("punct.psound");
  push_stroke((130,390-15*mincho)..(500,390)..(870,390+10*mincho),
    (0.7,3.3)--(2,2)--(0.7,3.3));
  expand_pbox;
enddef;

vardef punct.make_period(expr cpos) =
  push_stroke(fullcircle scaled (tsu_punct_size*1.15) shifted cpos,
    (2,2)--(2,2)--(2,2)--(2,2)--cycle);

  if tsu_pbrush_size>=30:
    set_bosize(0,40);
    push_lcblob(get_strokep(0));
  else:
    set_bosize(0,80);
  fi;

  push_pbox_explicit("punct.make_period",
    identity shifted (-0.5,-0.5) scaled (tsu_punct_size*1.5) shifted cpos);
```

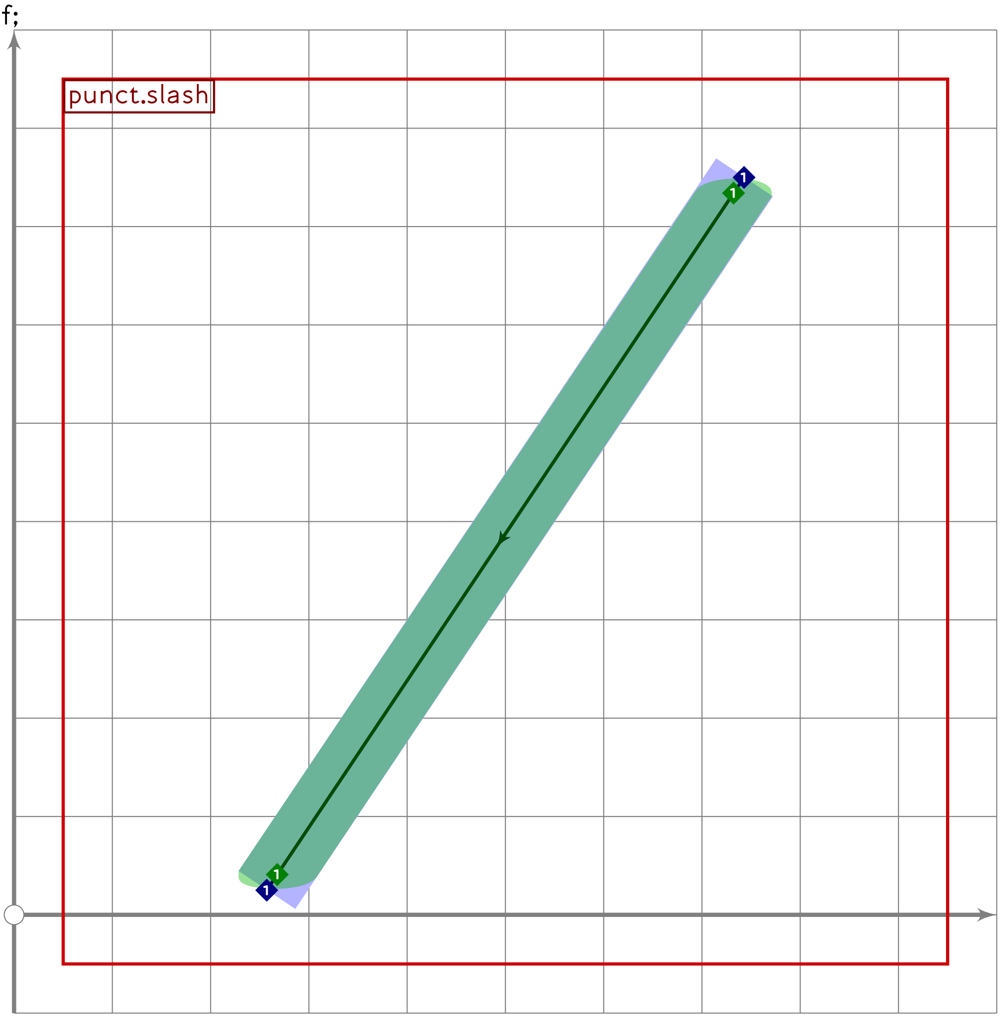
```
enddef;
```



```

vardef punct.section =
  push_pbox_toexpand(”punct.section”);

  (x1+x3)/2=x2=x4=x6=500;
  x5=0.8[x2,x1];
  2*(x5-x2)=0.45*(latin_wide_high_r-latin_wide_low_r);

  y1=y3=0.8[ypart centre_pt,latin_wide_high_r];
  y2=latin_wide_high_r;
  y4=0.35[ypart centre_pt,latin_wide_high_r];
  y5=ypart centre_pt;
  y4-y5=y5-y6;

  push_stroke((z1..z2..z3..z4..z5..z6) rotatedaround (centre_pt,-6),
    (1.8,1.8)—(1.2,1.2)—(1.7,1.7)—(1.3,1.3)—(2,2)—(1.5,1.5));

  push_stroke(get_strokep(0) rotatedaround (centre_pt,180),get_strokeq(0));
```

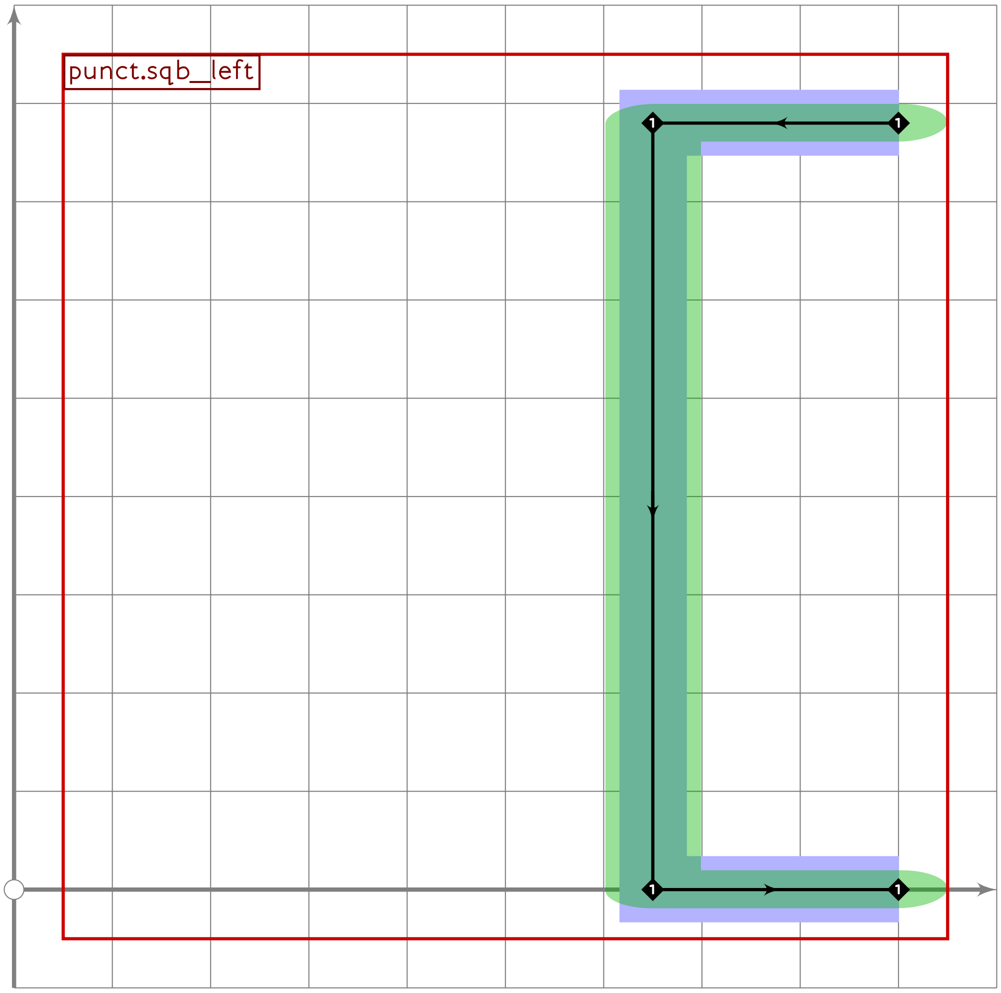
U+FF0F
tsuku.uniFF0F

```
  expand_pbox;
enddef;
```



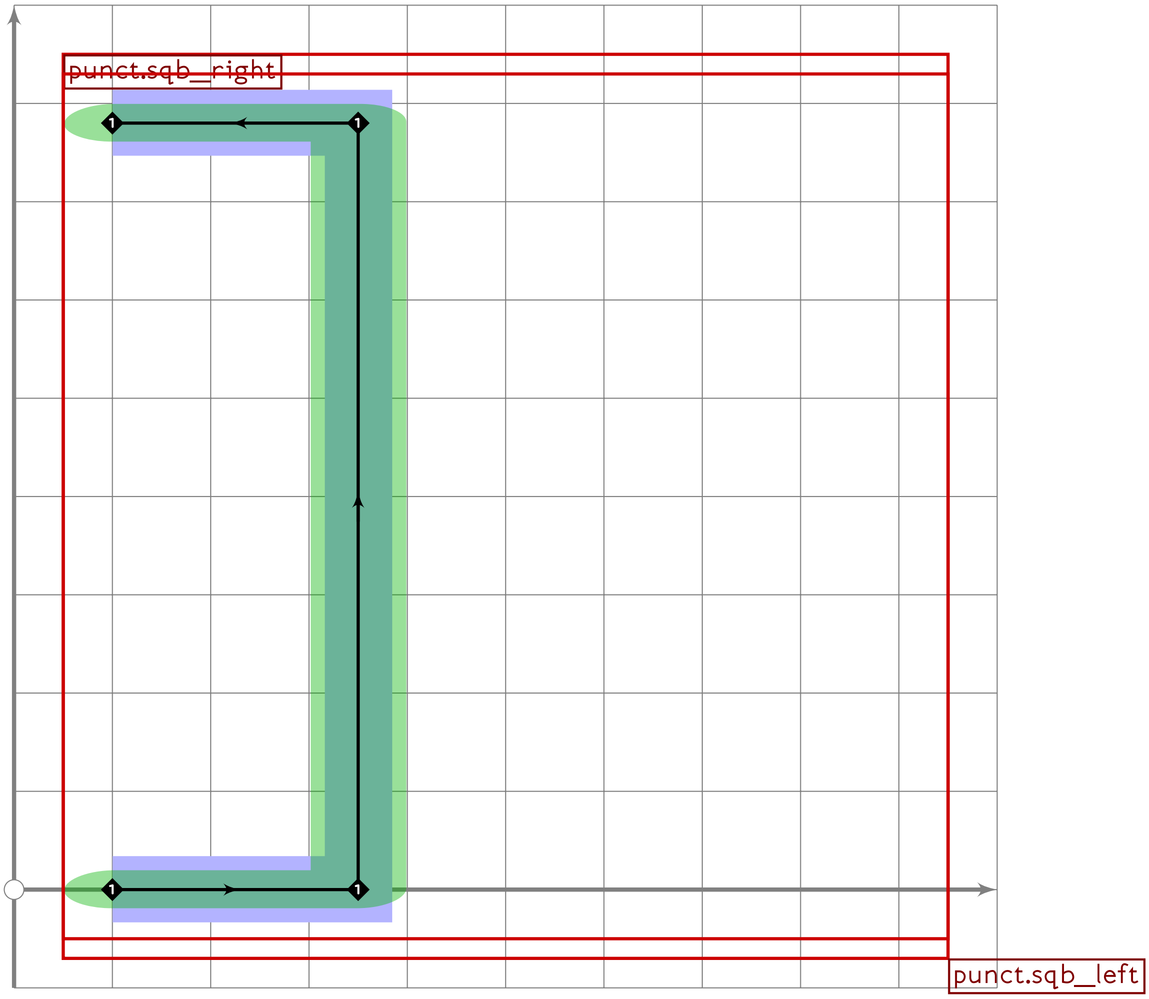
```

vardef punct.slash =
  push_pbox_toexpand(”punct.slash”);
  (x1+x2)/2=500;
  (x1-x2)=0.67(y1-y2);

  y1=latin_wide_high_v;
  y2=latin_wide_low_v;

  push_stroke((z1–z2)
      shifted -centre_pt scaled (tsu_punct_size/100) shifted centre_pt,
    (1.6,1.6)–(1.6,1.6));
  expand_pbox;
enddef;
```

U+FF3B
tsuku.uniFF3B



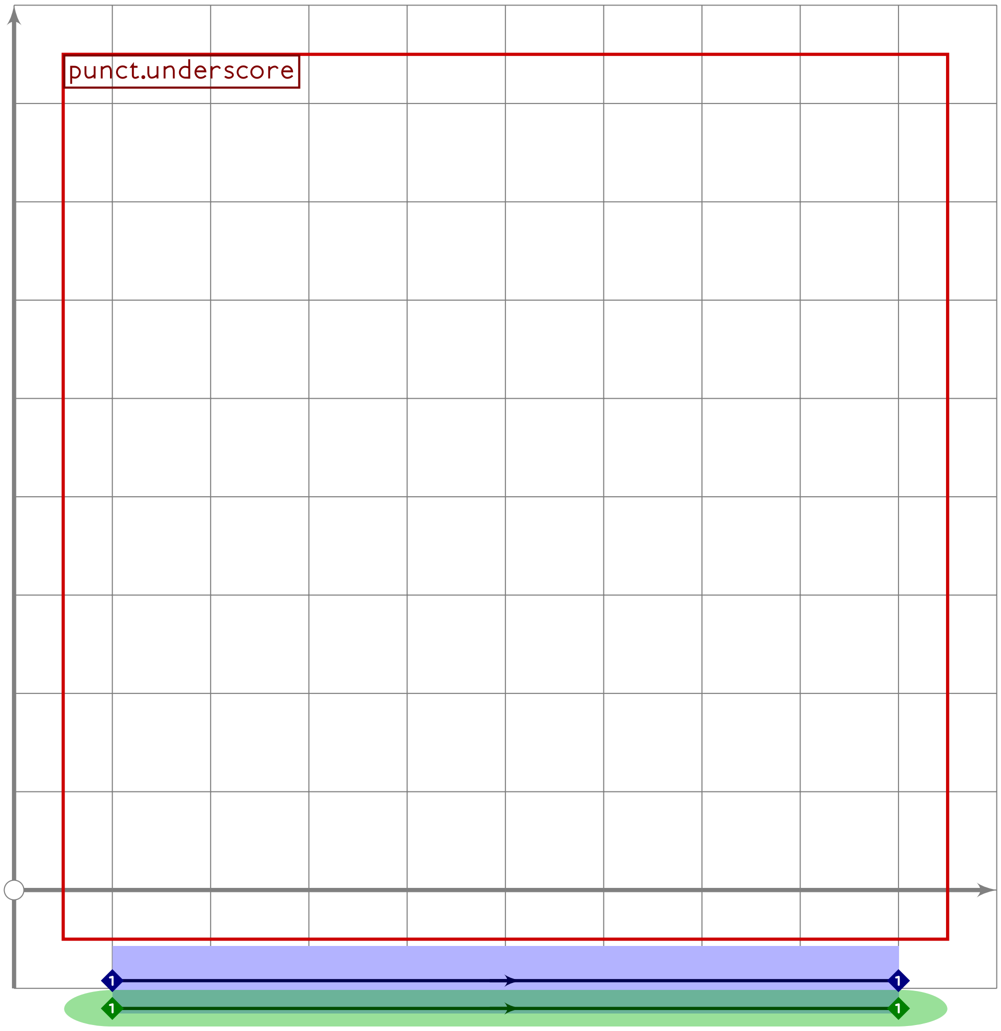
```
vardef  punct.sqb_left  =
   push_pbox_toexpand(”punct.sqb_left”);
   push_stroke((900,780)–
         (900-2.5*tsu_punct_size,780)–
         (900-2.5*tsu_punct_size,0)–
         (900,0),
      (2,2)–(2,2)–(2,2)–(2,2));
   set_bosize(0,90);
   set_botip(0,1,1);
   set_botip(0,2,1);
   expand_pbox;
enddef;
```

U+FF3D
tsuku.uniFF3D



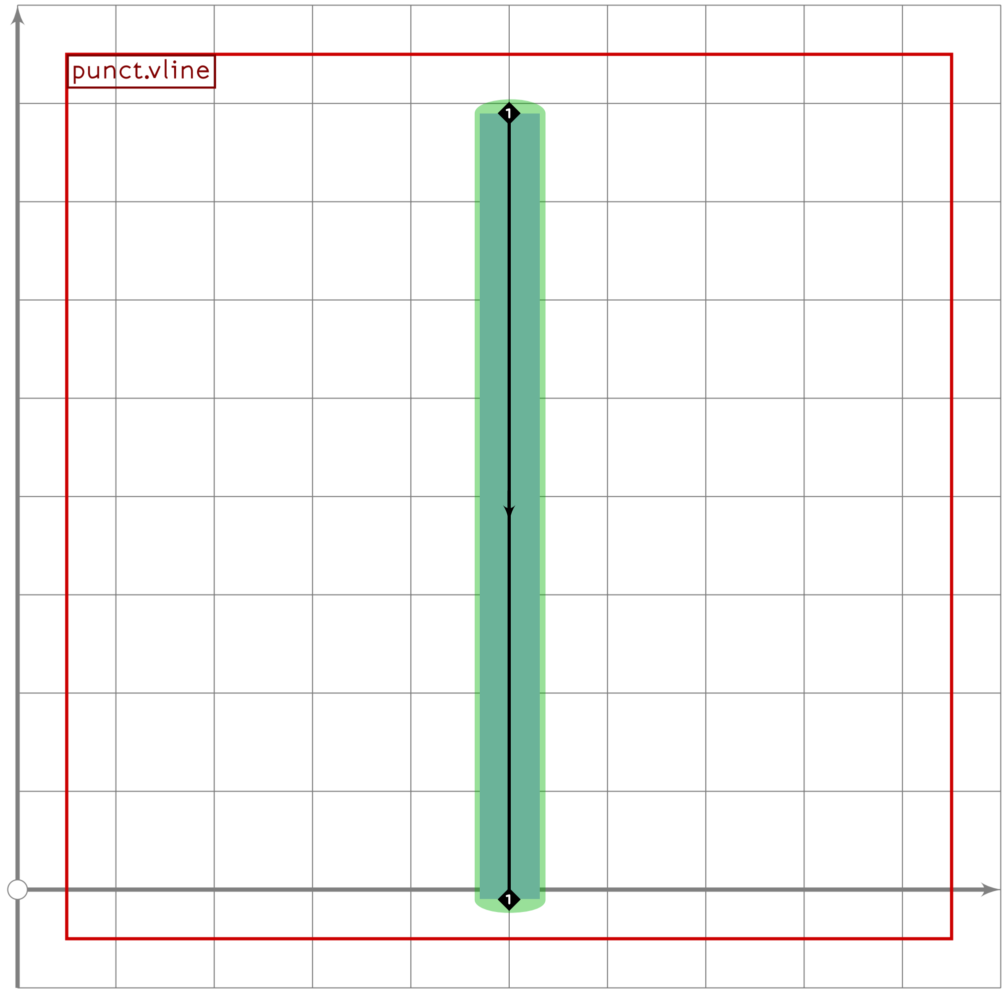
```

vardef punct.sqb_right =
  push_pbox_toexpand(”punct.sqb_right”);
  tsu_xform(identity rotatedaround (centre_pt,180))
    (punct.sqb_left);
  expand_pbox;
enddef;

vardef punct.times(expr t) =
  push_stroke(((-1,-1)--(1,1)) transformed t,(2,2)--(2,2));
  set_bosize(0,90);
  push_stroke(((-1,1)--(1,-1)) transformed t,(2,2)--(2,2));
  set_bosize(0,90);

  push_pbox_explicit(”punct.times”,
    identity shifted (-0.5,-0.5) scaled 2.4 transformed t);
enddef;
```

U+FF3F
tsuku.uniFF3F

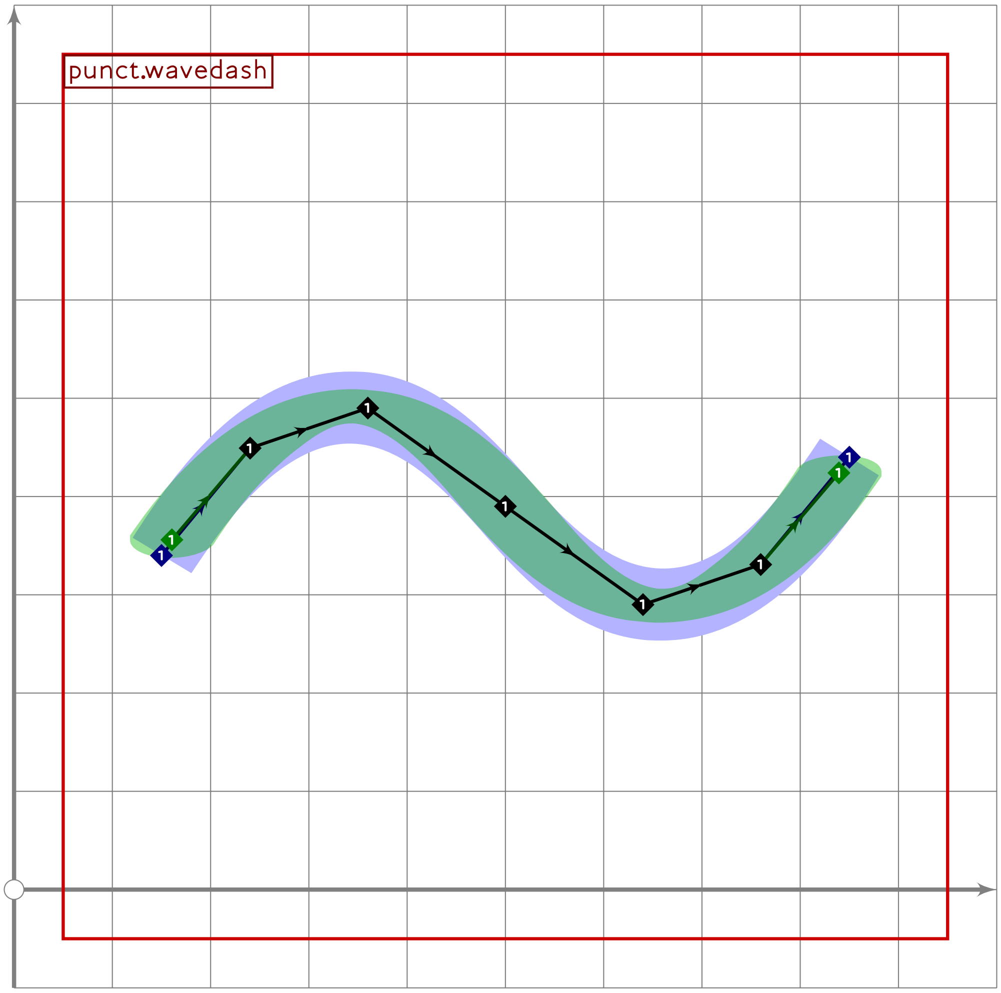


```

vardef punct.underscore =
  push_pbox_toexpand(”punct.underscore”);
  push_stroke((100,0.3[latin_wide_desc_h,latin_wide_low_h])–
      (900,0.3[latin_wide_desc_h,latin_wide_low_h]),
    (2,2)–(2,2));
  set_bosize(0,90);
  expand_pbox;
enddef;
```

U+FF5C
tsuku.uniFF5C

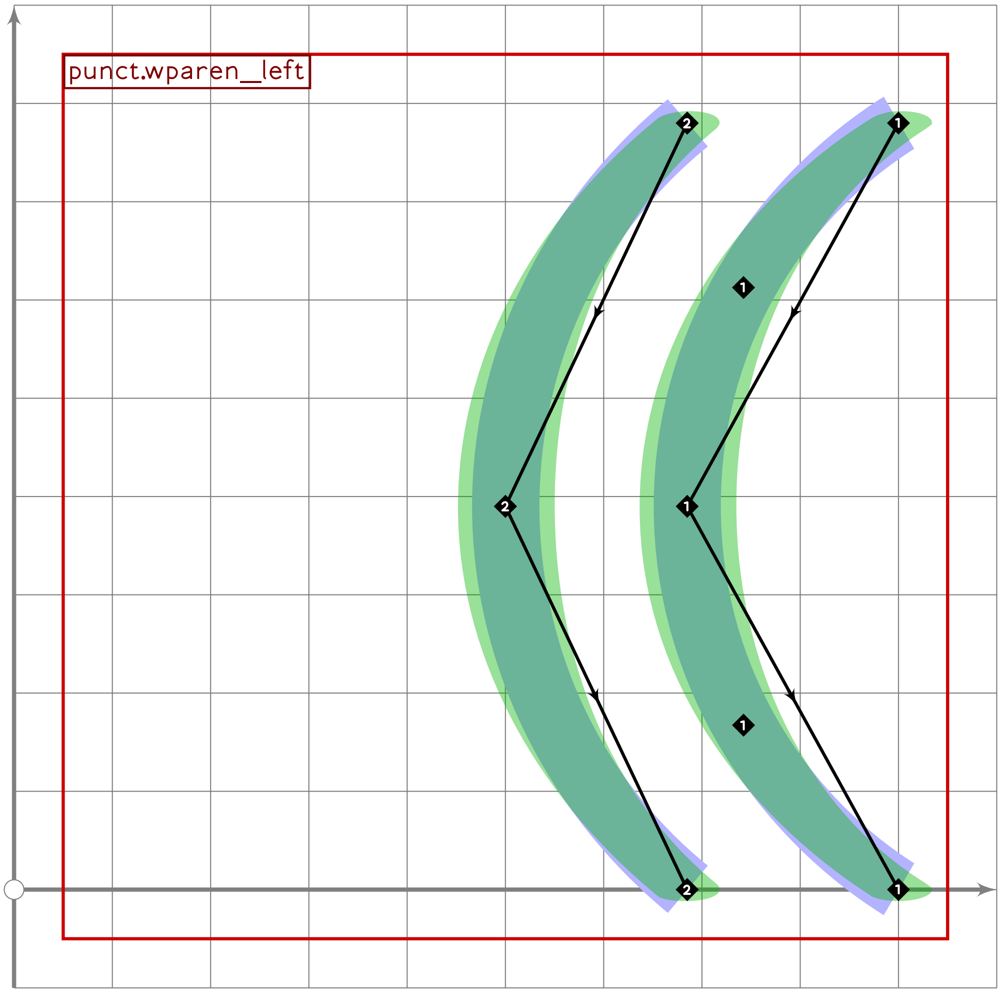


```
vardef  punct.vline =
   push_pbox_toexpand(”punct.vline”);
   push_stroke((500,690+tsu_punct_size)--(500,90-tsu_punct_size),
      (1.6,1.6)--(1.6,1.6));
   set_bosize(0,90);
   expand_pbox;
enddef;
```



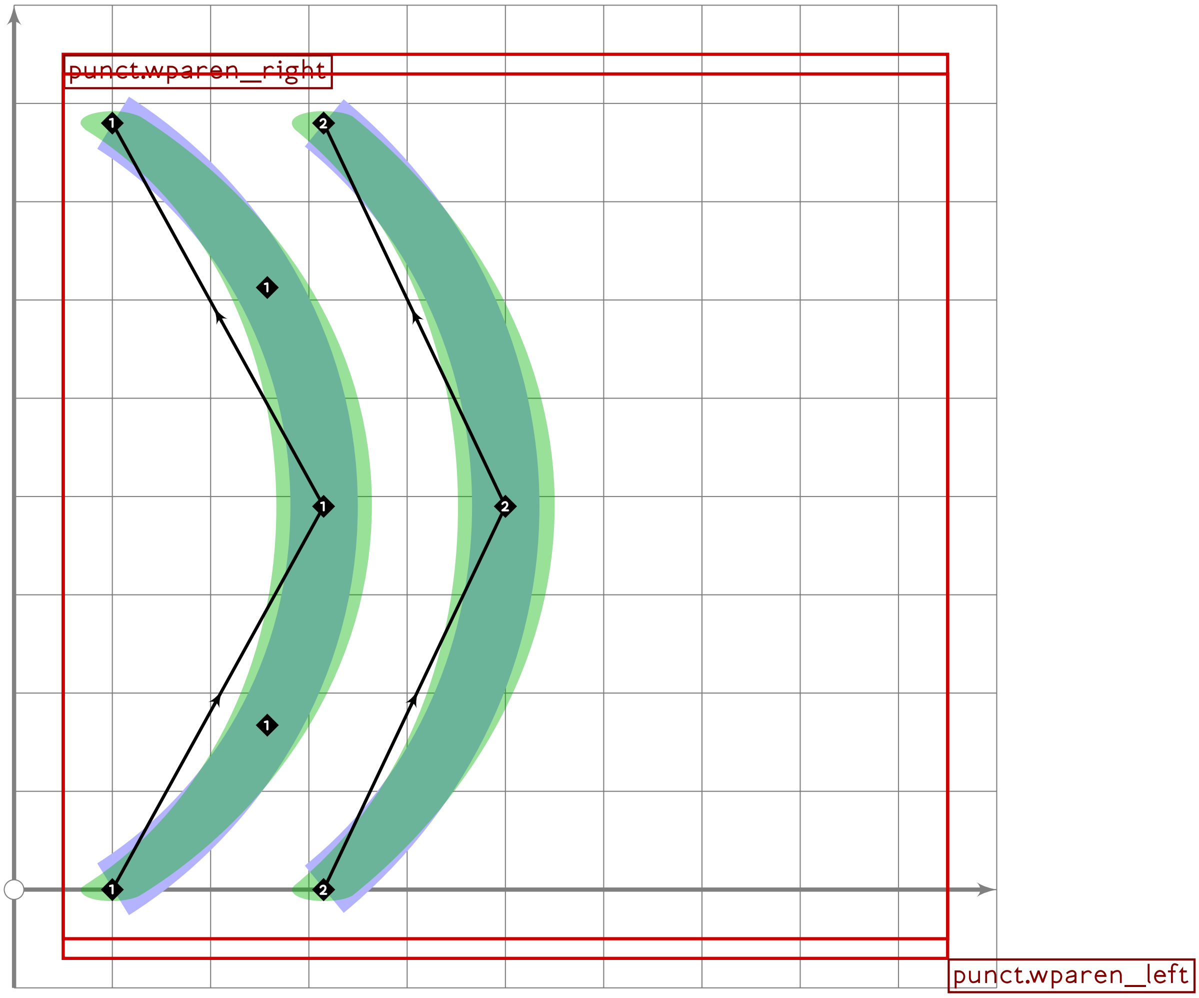
```
vardef punct.wavedash =
  push_pbox_toexpand(”punct.wavedash”);
  push_stroke(((-3.5,-0.5){curl 0}..(-1.4,1)..(0,0)..(1.4,-1)..
      {curl 0}(3.5,0.5)) scaled tsu_punct_size shifted centre_pt,
    (0.7,2.7)–(1.7,1.7)–(1.7,1.7)–(1.7,1.7)–(1.7,1.7)–
      (1.7,1.7)–(0.7,2.7));
  replace_strokep(0)(insert_nodes(oldp)(0.5,3.5));
  expand_pbox;
enddef;
```

U+FF5F
tsuku.uniFF5F



```
vardef  punct.wparen_left  =
  push_pbox_toexpand(”punct.wparen_left”);

  push_stroke((900,780)..(900-2.15*tsu_punct_size,390)..(900,0),
    (1.5,1.5)--(2,2)--(1.5,1.5));
  set_bosize(0,90);

  push_stroke((900-2.15*tsu_punct_size,780)..
        (900-4*tsu_punct_size,390)..
        (900-2.15*tsu_punct_size,0),
    (1.5,1.5)--(2,2)--(1.5,1.5));
  set_bosize(0,90);
  expand_pbox;
enddef;
```

U+FF60
tsuku.uniFF60



```
vardef punct.wparen_right =
  push_pbox_toexpand(”punct.wparen_right”);
  tsu_xform(identity rotatedaround (centre_pt,180))
    (punct.wparen_left);
  expand_pbox;
enddef;
```

serif.mp

```
%
% Serifs for Tsukurimashou
% Copyright (C) 2011, 2012 Matthew Skala
%
[Standard copyright notice]

inclusion_lock(serif);



% figure out size of brush
(sbrush_width,sbrush_height)=urcorner (
   fullcircle yscaled tsu_brush_shape
   rotated tsu_brush_angle
);
if sbrush_width>sbrush_height:
  sbrush_long:=sbrush_width;
  sbrush_short:=sbrush_height;
else:
  sbrush_short:=sbrush_width;
  sbrush_long:=sbrush_height;
fi;



vardef tsu_serif.latin.lrcore(expr bst,plp,dlp,l,bts,bos) =
  serif:=serif xscaled sbrush_long yscaled sbrush_short;
  serif:=serif xscaled ((1+3*xxpart tsu_rescaling_xf)/4)
    yscaled bos scaled bts;
  glstk[ngls]:=regenerate(serif shifted plp);
  ngls:=ngls+1;
enddef;

vardef tsu_serif.latin.left(expr bst,plp,dlp,l,bts,bos) =
  begingroup
    save serif;
    path serif;
    if sharp_corners:
      serif:=(-serif_size,1)--(-serif_size,-1)--
        (0,-1)--(0,1)--cycle;
    else:
      serif:=(-serif_size,1){left}..{right}(-serif_size,-1)--
        (0,-1)--(0,1)--cycle;
    fi;
    tsu_serif.latin.lrcore(bst,plp,dlp,l,bts,bos);
  endgroup;
```

```
enddef;

vardef tsu_serif.latin.right(expr bst,plp,dlp,l,bts,bos) =
  begingroup
    save serif;
    path serif;
    if sharp_corners:
      serif:=(0,1)--(0,-1)--
        (serif_size,-1)--(serif_size,1)--cycle;
    else:
      serif:=(0,1)--(0,-1)--
        (serif_size,-1){right}..{left}(serif_size,1)--cycle;
    fi;
    tsu_serif.latin.lrcore(bst,plp,dlp,l,bts,bos);
  endgroup;
enddef;

vardef tsu_serif.latin.leftright(expr bst,plp,dlp,l,bts,bos) =
  begingroup
    save serif,xoffs;
    path serif;
    if sharp_corners:
      serif:=(-serif_size,1)--(-serif_size,-1)--
        (serif_size,-1)--(serif_size,1)--cycle;
    else:
      serif:=(-serif_size,1){left}..{right}(-serif_size,-1)--
        (serif_size,-1){right}..{left}(serif_size,1)--cycle; fi;
    numeric xoffs;
    xoffs=xpart (dlp/abs(dlp));
    tsu_serif.latin.lrcore(bst,
      plp+if l=0: right else: left fi*50*sbrush_long*xoffs,
      dlp,l,bts,bos);
  endgroup;
enddef;

```

```

vardef tsu_serif.mincho.corner(expr bst,plp,dlp,l,bts,bos) =
  begingroup
    save serif;
    path serif;
    serif:=(-1,-0.3)..(0.25,-1.3)..(1,-1.2)..tension 1.2..
      (1,0.6)..(-0.25,1.3)..(-1,1.2)..tension 1.2..cycle;
    serif:=serif yscaled sqrt(tsu_brush_shape) rotated tsu_brush_angle
      scaled (bts*0.43*mincho_blob_size);
    glstk[ngls]:=regenerate(serif shifted plp);
    ngls:=ngls+1;
  endgroup;
```

```
enddef;

vardef tsu_serif.mincho.ulpoint(expr bst,plp,dlp,l,bts,bos) =
  begingroup
    save serif;
    path serif;
    serif:=(-1.5,2.7)..tension 2..(-1,0)..(-0.4,-0.3)..tension 1.5..
      (0.707,0)..(0.2,0.6)..(-0.1,1.1)..tension 2..(-1.2,2.9)..cycle;
    serif:=serif yscaled sqrt(tsu_brush_shape) rotated tsu_brush_angle
      scaled (bts*0.5*mincho_blob_size);
    glstk[ngls]:=regenerate(serif shifted
       (plp+(-1,0)*0.25*bts*(xpart dlp/abs(dlp))));
    ngls:=ngls+1;
  endgroup;
enddef;

vardef tsu_serif.mincho.triangle(expr bst,plp,dlp,l,bts,bos) =
  begingroup
    save serif;
    path serif;
    serif:=(-1.2,0)..(0,-0.8)..(1.2,0.4)..tension 2..(0.2,1.3)..
      (-0.2,1.3)..tension 2..cycle;
    serif:=serif yscaled sqrt(tsu_brush_shape) rotated tsu_brush_angle
      scaled (bts*0.5*mincho_blob_size);
    glstk[ngls]:=regenerate(serif shifted (plp+0.25*bts*dlp/abs(dlp)));
    ngls:=ngls+1;
  endgroup;
enddef;

vardef tsu_serif.mincho.llpoint(expr bst,plp,dlp,l,bts,bos) =
  begingroup
    save serif;
    path serif;
    serif:=(-2.1,-1.9)..(-1.8,-2.1)..tension 2..(-0.1,-1.2)..(0.2,-1.1)..
      (0.707,0.707)..(-1,-0.3)..tension 2..cycle;
    serif:=serif yscaled sqrt(tsu_brush_shape) rotated tsu_brush_angle
      scaled (bts*0.5*mincho_blob_size);
    glstk[ngls]:=regenerate(serif shifted plp);
    ngls:=ngls+1;
  endgroup;
enddef;

vardef tsu_serif.mincho.lpoint(expr bst,plp,dlp,l,bts,bos) =
  begingroup
    save serif;
    path serif;
    serif:=(-1.5,1.7)..tension 2..(-1,0)..(-0.4,-0.3)..tension 1.8..
      (0.707,0.2)..(0.2,0.8)..(-0.4,1.1)..tension 2..(-1.2,2.0)..cycle;
    serif:=reverse serif reflectedabout ((-1,1),(1,-1))
      xyscaled (0.7,0.9) shifted (0.3,-0.3);
    serif:=serif yscaled sqrt(tsu__brush__shape) rotated tsu__brush__angle
      scaled (bts*mincho__blob__size);
    glstk[ngls]:=regenerate(serif shifted plp);
    ngls:=ngls+1;
  endgroup;
enddef;

vardef tsu__serif.mincho.ktriangle(expr bst,plp,dlp,l,bts,bos) =
  begingroup;
    save serif,x,y,t,q;
    path serif;
    transform t;
    (0,0) transformed t=(0,0);
    right transformed t=(dlp/abs(dlp));
    z1=(dlp/abs(dlp)) rotated 90;
    if y1<-x1: x1:=-x1; y1:=-y1; fi;
    up transformed t=z1;
    q1:=2;
    q2:=q1+-+0.5;
    q3:=0.5*q2/q1;
    q5:=0.5+-+q3;
    z2=(0,q1);
    z3=(-q3,q5);
    z4=(q3,q5);
    serif:=((0.1[z2,z4]){z2-z4}..{z3-z2}(0.1[z2,z3])--
      (0.1[z3,z2]){z3-z2}..{z4-z3}(0.1[z3,z4])--
      (0.1[z4,z3]){z4-z3}..{z2-z4}(0.1[z4,z2])--cycle)
      shifted (0,-0.19)
      transformed t yscaled tsu__brush__shape
      rotated tsu__brush__angle
      scaled (bts*bos*0.94*mincho__blob__size) shifted plp;
    if (xxpart t)*(yypart t)<0: serif:=reverse serif; fi;
    glstk[ngls]:=regenerate(serif);
    ngls:=ngls+1;
  endgroup;
enddef;

vardef tsu__serif.mincho.khellipse(expr bst,plp,dlp,l,bts,bos) =
  begingroup;
    save serif,x,y,t,q;
    path serif;
    transform t;
    (0,0) transformed t=(0,0);
    right transformed t=(dlp/abs(dlp));
    z1=(dlp/abs(dlp)) rotated 90;
    up transformed t=z1;
    t:=t rotated -tsu_brush_angle yscaled tsu_brush_shape
      rotated tsu_brush_angle scaled (bts*bos*0.99*mincho_blob_size)
      shifted plp;
    z2=(0.5[x3,x4],1);
    if l=0:
      z3=(-0.5,0);
      z4=(1.3,0);
    else:
      z3=(-1.3,0);
      z4=(0.5,0);
    fi;
    serif:=((subpath (0.2,2) of (z4{up}..z2..{down}z3))--cycle) transformed t;
    glstk[ngls]:=regenerate(serif);
    ngls:=ngls+1;
  endgroup;
enddef;

% standard serif codes:
% 1 - Latin left
% 2 - Latin right
% 3 - Latin left and right
% 4 - Mincho corner blob
% 5 - Mincho blob, pointy to ul
% 6 - Mincho blob, triangular
% 7 - Mincho blob, point to ll
% 8 - Mincho blob, point to l
% 9 - Mincho kanji triangle
% 10 - Mincho kanji half-ellipse

boolean tsu_do_serif[];

vardef tsu_serif.standard(expr bst,plp,dlp,l,bts,bos) =
  if known tsu_do_serif[1]:
    if tsu_do_serif[1] and (bst=1):
      tsu_serif.latin.left(bst,plp,dlp,l,bts,bos);
    fi;
  fi;
  if known tsu_do_serif[2]:
    if tsu_do_serif[2] and (bst=2):
      tsu_serif.latin.right(bst,plp,dlp,l,bts,bos);
    fi;
  fi;
  if known tsu_do_serif[3]:
    if tsu_do_serif[3] and (bst=3):
      tsu_serif.latin.leftright(bst,plp,dlp,l,bts,bos);
    fi;
  fi;
  if known tsu_do_serif[4]:
    if tsu_do_serif[4] and (bst=4):
      tsu_serif.mincho.corner(bst,plp,dlp,l,bts,bos);
    fi;
  fi;
  if known tsu_do_serif[5]:
    if tsu_do_serif[5] and (bst=5):
      tsu_serif.mincho.ulpoint(bst,plp,dlp,l,bts,bos);
    fi;
  fi;
  if known tsu_do_serif[6]:
    if tsu_do_serif[6] and (bst=6):
      tsu_serif.mincho.triangle(bst,plp,dlp,l,bts,bos);
    fi;
  fi;
  if known tsu_do_serif[7]:
    if tsu_do_serif[7] and (bst=7):
      tsu_serif.mincho.llpoint(bst,plp,dlp,l,bts,bos);
    fi;
  fi;
  if known tsu_do_serif[8]:
    if tsu_do_serif[8] and (bst=8):
      tsu_serif.mincho.lpoint(bst,plp,dlp,l,bts,bos);
    fi;
  fi;
  if known tsu_do_serif[9]:
    if tsu_do_serif[9] and (bst=9):
      tsu_serif.mincho.ktriangle(bst,plp,dlp,l,bts,bos);
    fi;
  fi;
  if known tsu_do_serif[10]:
    if tsu_do_serif[10] and (bst=10):
      tsu_serif.mincho.khellipse(bst,plp,dlp,l,bts,bos);
    fi;
  fi;
enddef;

vardef tsu_serif.choose(expr bst,plp,dlp,l,bts,bos) =
  tsu_serif.standard(bst,plp,dlp,l,bts,bos);
enddef;
```