Creating cuneiform fonts with MetaType1 and FontForge

Karel Píška
Institute of Physics of the ASCR, v. v. i.
CZ-182 21 Prague, Czech Republic
piska (at) fzu dot cz

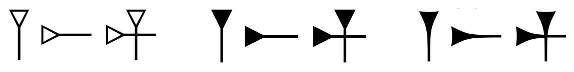
Abstract

A cuneiform font collection covering Akkadian, Ugaritic and Old Persian glyph subsets (about 600 signs) has been produced in two steps. With MetaType1 we generate intermediate Type 1 fonts, and then construct OpenType fonts using FontForge. We describe cuneiform design and the process of font development.

1 Introduction

I am interested in scripts, alphabets, writing systems, and in fonts, their computer representation. Ten years ago I decided to create Type 1 fonts for cuneiform, and last year, to extend them to Unicode OpenType versions. In my older Type 1 version (1998/9) [4] the raw text was written 'by hand' and then directly compiled into Type 1 with `t1asm` from `t1utils` [10]. The glyph set consisted of several separate Type 1 components to cover Akkadian (according to Labat) in a 'Neo-Assyrian' form (three files), Ugaritic, and Old Persian.

Three books served as principal sources: two Akkadian syllabaries edited by R. Labat [1] and F. Thureau-Dangin [2], and the encyclopedia *The World's Writing Systems* [3]. No scanning of pictures or clay tablets was performed. The fonts are based on a starting point — my simple design of *wedges* in three variant forms (see also Fig. 1, below):



A 'Academic' B 'Bold (Filled)' C 'Classic'

Our aim is to use free and open source software to produce "open source fonts". Thus, to create the fonts only non-proprietary tools have been employed: MetaType1; FontForge; `t1utils`, `gawk`; other standard Unix utilities such as bash, sed, sort, ... In the following sections we will explain the process of creating fonts and illustrate it with numerous examples.

2 Producing Type 1 with MetaType1

The MetaType1 package [6], developed by the authors of Latin Modern, TEX Gyre and other font collections (B. Jackowski, J. Nowacki, P. Strzelczyk):

- runs METAPOST (any available version) to produce `eps` files with outlines for all glyphs;
- collects all the data into one Type 1 file.

The information about the font and its glyphs is described in the METAPOST source files; additional macros are defined in MetaType1 extensions or may be appended by the user. For illustrations see the examples below. An explanation of some technical details and techniques how to work with MetaType1 can be found in the tutorial written by Klaus Höppner [7], which also includes a simple complete example and `Makefile`.

2.1 Font description in MetaType1

As usual with METAFONT or METAPOST the compilation is invoked by a main control file — `naakc.mp`:

```
input fontbase;
use_emergency_turningnumber;
input naak.mpe;
maybeinput "naakc.mpd";
maybeinput "naakc.mph";
maybeinput "naug.mph";
maybeinput "naop.mph";
beginfont
maybeinput "naak.mpg";
maybeinput "naug.mpg";
maybeinput "naop.mpg";
endfont
```

Global font parameters may be defined in a font header file — `naakc.mph`:

```
% FONT INFORMATION
pf_info_familyname "NeoAssyrianClassicType1";
pf_info_weight "Medium";
pf_info_fontname "NeoAssyrianClassicType1";
pf_info_version "002.001";
pf_info_author "Karel Piska at fzu.cz 2008";
pf_info_italicangle 0;
pf_info_underline -100, 50;
pf_info_fixedpitch false;
pf_info_adl 750, 250, 0;
italic_shift:=0;
```

Internal glyph names and metric data can be assigned as follows:

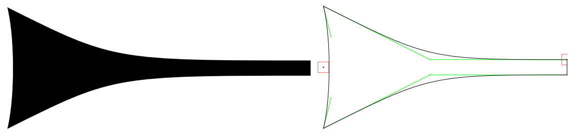
```
% INTRODUCE CHARS
standard_introduce("ash.akk");
standard_introduce("hal.akk");
standard_introduce("mug.akk");
standard_introduce("zadim.akk");
standard_introduce("ba.akk");
standard_introduce("zu.akk");
.....
% METRICS
wd._ash.akk=240; ht._ash.akk=160; dp._ash.akk=0;
wd._hal.akk=340; ht._hal.akk=160; dp._hal.akk=0;
wd._mug.akk=380; ht._mug.akk=220; dp._mug.akk=0;
wd._zadim.akk=460; ht._zadim.akk=220;
  dp._zadim.akk=0;
wd._ba.akk=420; ht._ba.akk=240; dp._ba.akk=0;
wd._zu.akk=500; ht._zu.akk=240; dp._zu.akk=0;
.....
```

(We do not define the encoding in Type 1.)

2.2 Glyph contours in MetaType1

Simple (atomic) elements — single wedges are defined by macros:

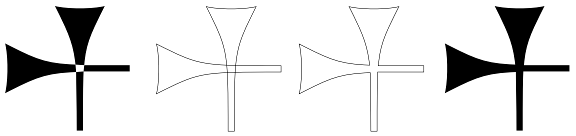
```
def wh(expr l,x,y)=
 r:=5; w:=40; b:=5; c:=20; d:=70;
 z[nw] 0=(x+l,y+r);
  z[nw] 0a=(x+d,y+r); z[nw] 1b=(x+d,y+r);
 z[nw] 1=(x,y+w);
  z[nw] 1a=(x+b,y+c); z[nw] 2b=(x+b,y-c);
 z[nw] 2=(x,y-w);
  z[nw] 2a=(x+d,y-r); z[nw] 3b=(x+d,y-r);
 z[nw] 3=(x+l,y-r);
 p[nw]=compose_path.z[nw](3);
 Fill p[nw];
 nw:=nw+1;
enddef;
```



Another definition of a wedge — a single path:

```
def pwh(expr l,x,y)=
 (x+l,y+r)..controls(x+d,y+r)..(x,y+w)
 ..controls(x+b,y+c)and(x+b,y-c)..(x,y-w)
 ..controls(x+d,y-r)..(x+l,y-r)--cycle;
enddef;
```

In compound elements, the rendering of intersecting areas may depend on printer/viewer. Therefore, removing overlap in Type 1 (and probably also in OpenType) is required. We use the macro `find_outlines`:

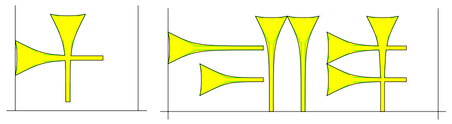


```
def whv(expr x,y)=
 save pa,pb,pc; path pa,pb,pc;
 r:=5; w:=40; b:=5; c:=20; d:=70;
 pa:=pwh(200,x,y); pb=pwv(200,x+80,y+100);
 find_outlines(pa,pb)(pc);
 p[nw]:=pc1;
 Fill p[nw];
 nw:=nw+1;
enddef;
```

Complete glyphs — that is, cuneiform signs — are demonstrated in the following examples. Horizontal/vertical wedges, composites or their groups are also defined by macros; the arguments, for example, denote their lengths and coordinates.

```
beginglyph(_mash.akk);
 save p; path p[]; nw:=0;
 whv(0,120);
 standard_exact_hsbw("mash.akk");
endglyph;
```

```
beginglyph(_sag.akk);
 save p; path p[]; nw:=0;
 wh(240,0,160);
 wh(160,80,80);
 wv(240,220,240);
 wv(240,300,240);
 whhv(400,120);
 standard_exact_hsbw("sag.akk");
endglyph;
```



2.3 Generating Type 1

An intermediate Type 1 is generated from scratch:

```
FN=$1  # font file name
MT1=$2 # MetaType1 direction
mpost '\generating:=0;' input $FN.mp
gawk -f $(MT1)/mp2pf.awk \
 -v CD=$(MT1)/pfcommon.dat -v NAME=$FN
gawk -f $(MT1)/packsubr.awk -v LEV=5 \
 -v OUP=$FN.pn $FN.p
t1asm -b $FN.pn $FN.pfb
```

with some simplifications (intentional for cuneiform): glyph design is simple; no kerning pairs are needed; the characters occupy independent boxes; no hyphenation; and no internal encoding in the intermediate Type 1 is defined. Theoretically, we could use a common Type 1 with several external encoding vectors, but in practice, joining all the glyphs into one OpenType font is a better and simpler solution.

3 FontForge and producing OpenType

Scripts in the FontForge scripting language read Type 1, build data for OpenType (especially, define the encoding) and then generate OTF and TTF files. Here is a table showing the Unicode areas with which we are concerned:

Definition of encoding (Unicode)

Plane 0	
U+0020–U+007F	ASCII block
Plane 1	
Cuneiform ranges	“Standard” Unicode
U+10380–U+1039F	Ugaritic
U+103A0–U+103D7	Old Persian
U+12000–U+123FF	Cuneiform signs
U+12500–U+1277F	Neo-Assyrian glyph container (temporary “Private Area”)

To begin, we introduce a new Unicode font:

```
#!/usr/bin/fontforge
# 1.sfd, 2.names, 3.pfb, 4.otf, 5.ttf
New();Reencode("UnicodeFull")
SetFontNames($2,$2,$2)
#SetFontOrder(3); # cubic
#SetFontOrder(2); # quadratic
ScaleToEm(250)
Save($1)
...
```

Then we copy glyphs from Type 1 to OpenType: we open and read a Type 1 font and access glyphs by name (in Type 1) and copy them to appropriate locations addressed by Unicode numbers:

```
Open($3);Select("ash.akk");Copy();Close();\
 Open($1);Select("u12501");Paste();
 Save($1);Close();
Open($3);Select("hal.akk");Copy();Close();\
 Open($1);Select("u12502");Paste();
 Save($1);Close();
Open($3);Select("mug.akk");Copy();Close();\
 Open($1);Select("u12503");Paste();
 Save($1);Close();
...
```

A FontForge user command eliminates the repetition:

```
#!/usr/bin/fontforge # copy.pe
# SF source font, SG source glyph
# DF destination font, DG dest. glyph
Open($1);Select($2);Copy();Close();
 Open($3);Select($4);Paste();
 Save($3);
```

with references for the Neo-Assyrian block:

```
# $SF is source font
# $SFD temporary font (internal)
./copy.pe $SF "ash.akk" $SFD u12501
./copy.pe $SF "hal.akk" $SFD u12502
./copy.pe $SF "mug.akk" $SFD u12503
...
```

The Neo-Assyrian glyphs are allocated in the container; existing glyphs are linked from the Cuneiform range by references:

```
Select("u12743");CopyReference();
 Select("u12000");Paste();
Select("u1264E");CopyReference();
 Select("u12009");Paste();
Select("u12580");CopyReference();
 Select("u1200A");Paste();
...
```

This operation may also be executed using a FontForge routine:

```
#!/usr/bin/fontforge # addref.pe
# 1. font, 2. glyph point in container
# 3. reference point
# addref.pe $fontname.sfd u12743 u12000
Open($1);Select($2);CopyReference();
Select($3);Paste();Save($1);
```

and then:

```
addref.pe $FN u12743 u12000
addref.pe $FN u1264E u12009
addref.pe $FN u12580 u1200A
...
```

Generating OpenType itself completes step 2: we can generate both OTF and TTF.

```
# $4 is OTF, $5 is TTF
Open($1);Generate($4); # with options
Open($1);Generate($5); # with options
```

Unfortunately, the glyph repertoire does not correspond to Unicode because, first, more than 300 glyphs do not have Unicode code points, and, on the other hand, my fonts cover only about 20% of the Unicode Sumerian-Akkadian cuneiform range (cuneiform signs and numeric signs).

In the final OpenType fonts, PostScript glyph names are omitted, the Akkadian glyph container (OTF/a) contains all Neo-Assyrian glyphs (according to Labat), partly defined by references in Unicode cuneiform block (OTF/c). Here is a table showing some of the correspondences:

PostScript	OTF/a	OTF/c
ash.akk	12501	u12038
hal.akk	12502	u1212C
mug.akk	12503	u1222E
zadim.akk	12504	
ba.akk	12505	u12040
zu.akk	12506	u1236A
su.akk	12507	u122E2
shun.akk	12508	
bal.akk	12509	u12044
adII.akk	1250A	
bulII.akk	1250B	
tar.akk	1250C	u122FB
an.akk	1250D	u1202D
ka.akk	1250F	u12157

3.1 Hinting

The OpenType output was “satisfactory” as autohinted with FontForge (Fig. 1); no hinting instructions are included in the TrueType fonts.

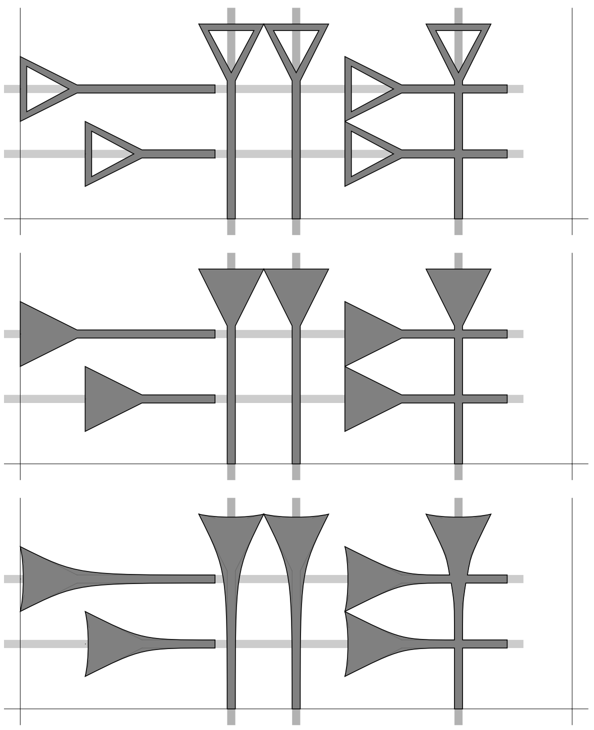


Figure 1: Wedge design in three variants with hinting.

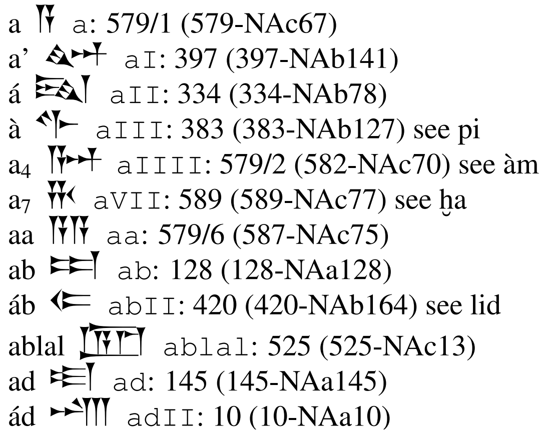
4 Support for (pdf)LaTeX and XeTeX

Old and simple LaTeX macros for Type 1 fonts and (pdf)LaTeX were modified for XeLaTeX to bind symbolic glyph names using their Unicode numbers.

```
\def\NAfontC#1{%         xakkadian.sty
 \font\NAC="[nacunc.ttf]" at #1pt
 }
\def\NAfont{\NAfontC{10}} % default font
%
\def\AKK{%
\NAfont%
\def\ash{{\NAC\char"12501}}%
\let\dil\ash\let\tilIIII\ash\let\ttil\ash%
\let\rum\ash\let\ruIII\ash\let\ina\ash%
\let\asIII\ash\let\azIII\ash%
\def\hal{{\NAC\char"12502}}\let\buluh\hal%
\def\mug{{\NAC\char"12503}}%
\let\muk\mug\let\muq\mug\let\puk\mug%
\def\zadim{{\NAC\char"12504}}%
\def\ba{{\NAC\char"12505}}%
\let\paII\ba%
\def\zu{{\NAC\char"12506}}%
\let\ssuII\zu%
...
```

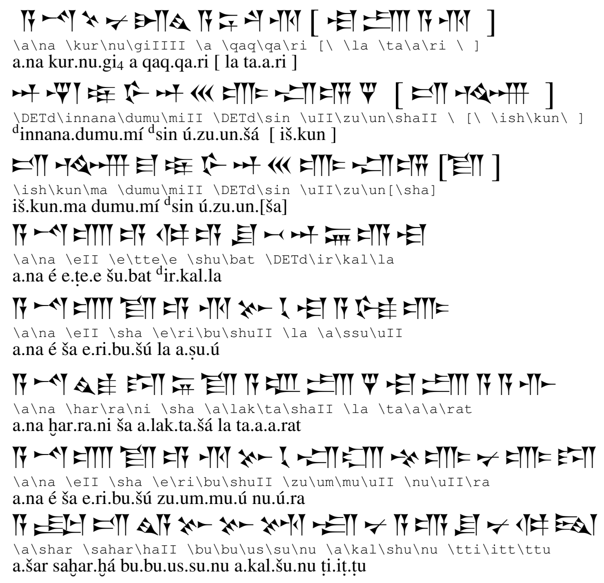
The two following examples show font usage.

1. Glyph index, numbers correspond to Labat [1]:



a `a`: 579/1 (579-NAc67)
a’ `aI`: 397 (397-NAb141)
á `aII`: 334 (334-NAb78)
à `aIII`: 383 (383-NAb127) see pi
a$_4$ `aIIII`: 579/2 (582-NAc70) see àm
a$_7$ `aVII`: 589 (589-NAc77) see ḫa
aa `aa`: 579/6 (587-NAc75)
ab `ab`: 128 (128-NAa128)
áb `abII`: 420 (420-NAb164) see lid
ablal `ablal`: 525 (525-NAc13)
ad `ad`: 145 (145-NAa145)
ád `adII`: 10 (10-NAa10)

2. Sample text in Akkadian (with transliteration):



`\a\na \kur\nu\giIIII \a \qaq\qa\ri [\ \la \ta\a\ri \ ]`
a.na kur.nu.gi$_4$ a qaq.qa.ri [la ta.a.ri]

`\DETd\innana\dumu\miII \DETd\sin \uII\zu\un\shaII \ [\ \ish\kun\ ]`
^{d}innana.dumu.mí ^{d}sin ú.zu.un.šá [iš.kun]

`\ish\kun\ma \dumu\miII \DETd\sin \uII\zu\un[\sha]`
iš.kun.ma dumu.mí ^{d}sin ú.zu.un.[ša]

`\a\na \eII \e\tte\e \shu\bat \DETd\ir\kal\la`
a.na é e.ṭe.e šu.bat ^{d}ir.kal.la

`\a\na \eII \sha \e\ri\bu\shuII \la \a\ssu\uII`
a.na é ša e.ri.bu.šú la a.ṣu.ú

`\a\na \har\ra\ni \sha \a\lak\ta\shaII \la \ta\a\a\rat`
a.na ḫar.ra.ni ša a.lak.ta.šá la ta.a.a.rat

`\a\na \eII \sha \e\ri\bu\shuII \zu\um\mu\uII \nu\uII\ra`
a.na é ša e.ri.bu.šú zu.um.mu.ú nu.ú.ra

`\a\shar \sahar\haII \bu\bu\us\su\nu \a\kal\shu\nu \tti\itt\ttu`
a.šar saḫar.ḫá bu.bu.us.su.nu a.kal.šu.nu ṭi.iṭ.ṭu

Line 1: Akkadian text using the cuneiform font
Line 2: The corresponding source input in the LaTeX command { `\AKK` *source* }
Line 3: Transliteration (dots and spaces added manually)

5 Conclusion

Both METAFONT and MetaType1 (=METAPOST) are programmable. But METAFONT produces only bitmaps, while in MetaType1, we must not define areas to fill or unfill with bitmap matrices which would depend on the device (resolution, blacker and other parameters). Rather, we are restricted to outlines:

- glyphs must be defined by closed curves, i.e. sequences of splines;
- we produce the Type 1 format directly;
- the MetaType1 commands `Fill/Unfill` denote the output of curves in the PostScript Type 1

representation with proper path direction and correct order of spline segments;

- final filling/unfilling is delegated to the PostScript/PDF rasterization systems.

Between PFB (Type 1) and OTF (PS/CFF flavored OpenType) we can find only formal differences in internal representation and organization; the mathematical outline curves and hints are identical. On the other hand, OTF and TTF (TrueType flavored OpenType) may differ in approximation of curve segments, since the underlying representations use cubic and quadratic polynomials, respectively. For our simple cuneiform design of wedges, though, a common approximation is workable. XƎLATEX can read all font formats: TEX fonts, OTF, TTF, etc.; OpenOffice 2.3 (on my computer) can work only with TTF. (I can say nothing about MS Word because I do not have this product.)

MetaType1 and FontForge give the advantage of programmability with open source data. In FontForge, the interactive approach in glyph design is dominant; theoretically we could define glyph outlines in the FontForge scripting language but it would be very difficult and inefficient. METAFONT/METAPOST (MetaType1) are more flexible and modular: they allow for solving mathematical equations, common processing and maintenance of related fonts, automatic calculation of parameters, and systematic modifications.

A typical task for MetaType1 is to combine a small number of components into many composite glyphs uniformly. This is common for "special kinds of fonts": just as Latin Modern and TEX Gyre can combine letters + accents, the cuneiform fonts can combine wedges; operations to produce composite glyphs can be defined and applied in a simple way, and generation and maintenance can be repeated for numerous fonts.

The older non-Unicode versions of cuneiform fonts have been already referenced in the subsection "External links / Fonts" in `http://en.wikipedia.org/wiki/Cuneiform_script` (a web search for "cuneiform" should find it also). They have been already used by scholars; e.g. for syllabaries and computer transliteration of sample texts for students.

Now I plan to finish and publish the new "Unicode" version, by extending the glyph repertoire to other glyphs and other shapes, corresponding to other languages and their historical period. Preliminary experimental OpenType fonts are available on my web site [11].

My final wish is that the MetaType1 package would be extended to "MetaOpenType" to produce OpenType font formats directly.

6 Acknowledgements

I want to thank the authors of MetaType1, FontForge (G. Williams) and other developers and maintainers of free and open source software.

References

[1] René Labat. *Manuel d'épigraphie akkadienne.* Troisième édition. Imprimerie nationale, Paris, 1959.

[2] F. Thureau-Dangin. *Le syllabaire accadien.* Librairie Orientaliste Paul Geuthner, Paris, 1926.

[3] *The World's Writing Systems.* P. T. Daniels and W. Bright, eds. Oxford University Press, New York–Oxford, 1996.

[4] Karel Píška. Fonts for Neo-Assyrian Cuneiform. *Proceedings of the EuroTEX Conference (Paperless TEX)*, Heidelberg, Germany, September 20–24, 1999, Günter Partosch and Gerhard Wilhelms, eds. Gießen, Augsburg, 1999, ISSN 1438-9959, 142–154. `http://www-hep.fzu.cz/~piska/cuneiform.html`

[5] Cuneiform script (Wikipedia). `http://en.wikipedia.org/wiki/Cuneiform_script`

[6] Bogusław Jackowski, Janusz M. Nowacki, Piotr Strzelczyk. Programming PostScript Type 1 fonts using MetaType1: Auditing, enhancing, creating. *Proceedings of EuroTEX 2003*, Brest, France, 24–27 June 2003. *TUGboat* 24:3, pp. 575–581, `http://tug.org/TUGboat/Articles/tb24-3/jackowski.pdf`; `CTAN:fonts/utilities/metatype1`; `ftp://bop.eps.gda.pl/pub/metatype1`.

[7] Klaus Höppner. Creation of a PostScript Type 1 logo font with MetaType1. *Proceedings of XVII European TEX Conference, 2007.* *TUGboat* 29:1, pp. 34–38, `http://tug.org/TUGboat/Articles/tb29-1/tb91hoeppner.pdf`.

[8] George Williams. Font creation with FontForge. *EuroTEX 2003 Proceedings*, *TUGboat* 24:3, pp. 531–544, `http://tug.org/TUGboat/Articles/tb24-3/williams.pdf`; `http://fontforge.sourceforge.net`.

[9] Free Software Foundation. GNU awk, `http://www.gnu.org/software/gawk`.

[10] Eddie Kohler. `t1utils` (Type 1 tools), `http://freshmeat.net/projects/t1utils`.

[11] `http://www-hep.fzu.cz/~piska/cuneiform/opentype.html`