# There is no end: Omega and Zapfino

*Of the making of books, there is no end.* — The Prophet Muhammad

William F. Adams

ATLIS Graphics
75 Utley Drive, Suite, 110
Camp Hill, PA 17011
email: `willadams@aol.com`

**Abstract**

The future of type is OpenType (Adobe and Microsoft's successor to Apple's "Royal" font technology which was licensed to Microsoft as TrueType), Unicode, and other extensions of TrueType and the Type 1 font format such as ATSUI (Apple Typographic System for Unicode Information).

Fortunately, for Unicode in TeX there is Omega which coupled with the other strengths of TeX can be sufficient to take advantage of new technologies without explicit support with the proper (or improper) techniques.

This paper will be an explanation and exploration of this, looking at a specific font and format (the `.dfont` ATSUI-enabled version of Zapfino), arguably very nearly a worst-case scenario, and how it can be dissassembled into individual glyphs and seamlessly stitched back together as an Omega Virtual Font with a matching Omega Translation Process to automatically insert ligatures and swash and variant forms using ASCII markup in an otherwise ordinary `.tex` source file which can then be used in a pre-press ready workflow.

## Introduction

Mac OS X derives from Nextstep, gaining support for Mac Resource /Suitcase fonts and PC truetype fonts, but losing support for Unix `.pfa` Next style .font bundles.

Mac OS X provides many of its system fonts in the new `.dfont` format, which while a straightforward storing of a Mac-style TrueType font in the file proper (the datafork in Mac parlance) instead of the resource fork as was done with Mac OS 9 and earlier is not equivalent to a PC format TrueType font stored in a `.ttf` file. Although there are programs which can open and parse fonts stored in a `.dfont` now (Pfaedit[1] is a notable example), my interpretation of Apple's licensing agreement leads me to believe that any such parsing or conversion would not be allowed by that license.

However, having purchased Mac OS X and its $10,000 worth of fonts, one cannot help but wish to use them. Although Zapfino works well in "Cocoa" programs in Mac OS X such as TextEdit.app, its special features such as ligatures are enabled by "Apple Advanced Typography" (AAT) which is unfortunately not supported by the more traditional "Carbon" Macintosh applications in which class at this writing all mainstream graphic design applications are.[2]

This is unfortunately quite limiting, either one must limit oneself to Cocoa applications, or in applications such as InDesign, make use of its Glyph palette to insert alternates and ligatures by repetitive pointing-and-clicking. Since there is no TeX variant which can access system fonts on Mac OS X as of this writing,[3] one must develop a work-around which allows one to access arbitrary fonts from within TeX and to simulate the capabilities of OpenType or Apple Advanced Typography. The large character sets of fonts such as Apple Chancery or Zapfino make accessing characters in 8-bit blocks untenable, so Omega is an obvious choice. This serves two purposes, first, it makes the typeface, Zapfino by Prof. Hermann Zapf

[1] Renamed to *FontForge*, this wonderful program is available from `http://pfaedit.sourceforge.net`.

[2] Since then, the opensource drawing program Cenon has been released for Mac OS X as well as OPENSTEP 4.2 and GNUstep. It is available from `http://www.cenon.info`.

[3] Jonathan Kew has since released XeTeX, a successor to his TeX/GX program for Apple's QuickDraw/GX which runs on Mac OS X making AAT fonts accessible. It is available from `http://scripts.sil.org/xetex`.

available for use in TEX by way of Omega, second, it provides an encoding scheme and mechanism to access arbitrary ligatures and alternates.

This then begs the question of how does one install a font into a program (system) which doesn't have direct support for that font format or its capabilities? The solution is blindingly simple in retrospect, consider what the system does support (PostScript by way of `dvips` and the `\special` mechanism) and where that intersects with the capabilities of systems which can use the font to its fullest (Encapsulated PostScript File graphics). The solution then is to load all of the characters of a font into a file so that they may then each be output as individual `.eps` files, stitch said files together as a virtual font and then rely on `dvips` to put everything back together. Zapfino however, has so many characters (1,417 in the version bundled with Mac OS X 10.2 "Jaguar"[4]) that Omega, with its support for Unicode which provides for large character sets is needed. Omega also affords the Omega Translation Process (OTP), which is far more efficient at enabling long ligatures than the standard TEX or PostScript mechanisms. Fortunately, `odvips` supports the aforementioned `special` mechanism as well.

Although font metric information is probably not protectable, there is no reasonable method at present to access the data stored within the Zapfino font file which wouldn't run afoul of Apple's license which forbids decompilation or other modification. Presumably a program using the nsText object could access such data on a per character basis and write that out in a useful format, but I'm a graphic design, not computer science major, so a copy of the `.afm` files provided by Volker Schaa (he had received a copy of the original Linotype Zapfino CD-ROM from Prof. Zapf as a gift) was used as a beginning point. These files were converted into standard `.tfm` files using `afm2tfm` and thence to `.pl` files using `tftopl`. The file for the font Zapfino One served as the basis for `zapfino.ovp` the base font file. The utility `ovp2ovf` was then used to create `ovf` and `ofm` files for Omega to use. The files are stored in `~/texmf/fonts/ovp`, `~/texmf/fonts/ovf` and `~/texmf/fonts/ofm` respectively.

Before testing could begin in earnest, it was necessary to have the letterforms themselves accessible to output. This was done by using Adobe InDesign to typeset an 'Adobe Tagged Text' file which enumerated all of the characters in Zapfino. First, a single character was set in the font Zapfino in InDesign at 72 points size with 96 points leading and then exported (File | Export... select 'Adobe InDesign Tagged Text' in the 'Formats' pop-up), yielding a file with the following line needed for our purposes:

```
<cTypeface:><cSize:><cLeading:><cFont:><cHang:><pHyphenationLadderLimit:><pHyphenation:>
<pHyphenationZone:><pTabRuler:><ParaStyle:><pHyphenationLadderLimit:0><pHyphenation:0>
<pHyphenationZone:0.000000><pTabRuler:
28.000000\,Left\,.\,0\,\;56.000000\,Left\,.\,0\,\;84.000000\,Left\,.\,0\,\;112.000000\,Left\,.
\,0\,\;140.000000\,Left\,.\,0\,\;168.000000\,Left\,.\,0\,\;196.000000\,Left\,.\,0\,
\;216.000000\,Left\,.\,0\,\;224.000000\,Left\,.\,0\,\;252.000000\,Left\,.\,0\,\;280.000000\,
Left\,.\,0\,\;308.000000\,Left\,.\,0\,\;336.000000\,Left\,.\,0\,\;>
<cTypeface:Regular><cSize:72.000000><cLeading:96.000000><cFont:Zapfino>
<cHang:Baseline>A<0xFFFD><cTypeface:><cSize:><cLeading:><cFont:><cHang:>
<cSpecialGlyph:><cTypeface:Regular><cSize:72.000000><cLeading:96.000000>
<cFont:Zapfino><cHang:Baseline><cNextXChars:Page>
```

(The exported character was "A" — there is some additional text above and below said line, but it need merely be preserved in its entirety for later use.) After a little study and experimentation, it was found that the placed character could be replaced with '`<cSpecialGlyph:####>`' where `####` was a number ranging from 1 (the first character, "A" shown in the Unicode glyph palette in in Mac OS X) to 1417 (the last character, the open Apple symbol), so an Excel file was created with 1,417 rows and three columns. The first column was everything before the "A" endlessly repeated, with '`<cSpecialGlyph:`' added. The second column incremented the current line number starting from 1. The third column closed out the line, adding in a '`>`' to close the `cSpecialGlyph` directive. This was exported as text and replaced the line shown above in the 'Adobe InDesign Tagged Text' file which was then saved.

Next, a template file was created, (File | New | Document... the 'Facing Pages' checkbox cleared, and the one for 'Master Text Frame' was checked, the 'Page Size' set to 'Letter', 'Orientation' to 'Landscape', 'Margins' and 'Columns' were left at their default) as shown in Figure 1. After clicking "OK" and getting a

[4] Since this writing, Linotype has released Zapfino Extra OpenType which provides even more characters, most notably small caps, and Forte which provides five additional weights. The technique here should also work with this new version. More information on Zapfino Extra is available from http://www.linotype.com/1897/linotypezapfinoextra-folder.html.

new document the “A-Master” page icon in the “Pages” palette (Window | Pages) is double-clicked to allow editing of the master text frame. The master text frame is then selected and set to the coordinates X: 43p6, Y: 31p6, W: 39p0 and H: 33p0 using the “Transform” palette (Window | Transform) as shown in Figure 2. The main document is then returned to by double-clicking on Page 1 in the ‘Pages’ palette. Clicking with the ‘Text’ tool in the text block, one then chooses File | Place... and navigates to the ‘Adobe InDesign Tagged Text’ file created above and places it in the document so that it auto-flows to create 1,417 pages with one character per page (click on the ‘Master Text Frame’ while holding down the `<Shift>` key). The file is then saved as `Zapfino-chars` in a convenient location.

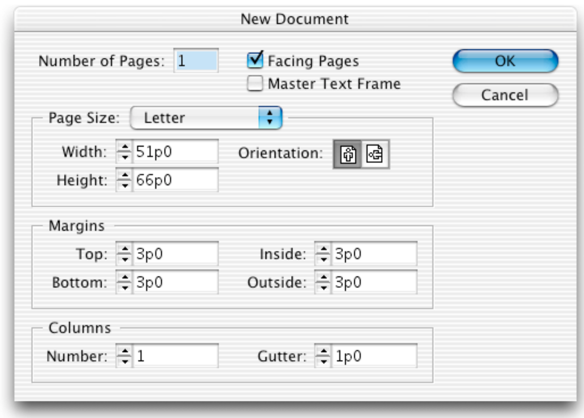

New Document
Number of Pages: 1
[✓ Facing Pages]
Master Text Frame
Page Size: Letter
Width: 51p0
Height: 66p0
Orientation:
Margins
Top: 3p0 Inside: 3p0
Bottom: 3p0 Outside: 3p0
Columns
Number: 1 Gutter: 1p0

**Figure 1**: Adobe InDesign ‘New Document’ Dialog

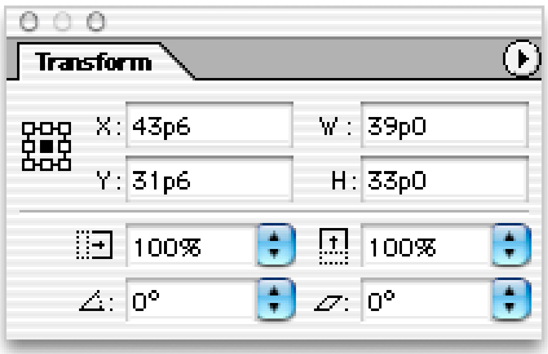

Transform
X: 43p6 W: 39p0
Y: 31p6 H: 33p0
100% 100%
0° 0°

**Figure 2**: Adobe InDesign ‘Transform’ Palette

Once one has a document with all of the desired characters, it is then a matter of exporting each page as a `.eps`. Fortunately, Adobe InDesign affords a menu option specifically for this, File | Export... which includes direct support for the `.eps` format. Choosing “EPS” in the Formats pop-up menu takes one to a dialogue box where one can select various settings. For the initial font the settings used were: ‘PostScript’: Level 2, ‘Color’: Gray, ‘Preview’: None, ‘Embed Fonts’: Subset, ‘Data Format’: ASCII. InDesign is able to subset fonts in such a way that individual subsetted fonts may be recombined seamlessly within a PostScript file or `.pdf` without any of the encoding conflicts sometimes seen in fonts subsetted by Adobe Acrobat or other programs. The ‘Data Format’ must be set to ASCII, since (o)dvips cannot handle a binary encoded `.eps` file created in this fashion. The files are exported to `~/Library/texmf/fonts/eps/Apple/Zapfino` for later usage.

Then the `.pl` file for Zapfino-One was used as a basis for the initial `zapfino.ovp` Omega Virtual Font Property List. Notable settings which were necessary included setting the font’s natural optical size `(DESIGNSIZE R 24)`. This technique is size-specific, and a different font must be made for each size which one wishes to typeset at. See the “Peace” below for an example of a work-around of this limitation.

With the character outlines now available, it is possible to place them within the virtual font using the `special` mechanism in odvips. Where each character has an entry like:

```
(MAP
   (SETCHAR O 353)
   )
```

This is replaced with something like:

```
(MAP
   (PUSH)
   (MOVELEFT R 1.129)
   (MOVEDOWN R 1.724)
   (SPECIAL PSfile=Zapfino-chars_277.eps
              hscale=13.28 vscale=13.28)
   (POP)
   (MOVERIGHT R .543)
   )
```

(the above is taken from the entry for “IJ”). First the current position is stored `(PUSH)`, then an adjustment is made for the offset of the character origin on the `.eps` file `(MOVELEFT)` and `(MOVEDOWN)`, the character is placed (`SPECIAL PSfile=Zapfino-chars_277.eps`, scaled `hscale` and `vscale`, then the previous position is restored (`POP`) and the position advanced to match the `CHARWD` (`MOVERIGHT R .543`) (where .543 is the width of the "IJ" character).

With all of this done, one can begin testing the font so as to check the metrics of the characters. A number of the characters in Zapfino were re-drawn between the original Type 1 format and the version Apple bundled with Jaguar, so this step could not be skipped. How to space and test a new typeface design, is well documented in several excellent references, most notably Stephen Moye's *Fontographer: Type by Design* and Walter Tracy's *Letters of Credit*, both of which are highly recommended to the aspiring type designer (or installer). In short, one sets various "standards" and other characters between them, adjusting the most common and easily spaced characters ("n" and "o") first, working toward those which are more difficult and assigning predetermined sidebearings to similar characters (so "m" gets the same sidebearings as "n"). Often when designing a new typeface, the initial attempt to set the sidebearings will result in a determination that certain characters must be re-drawn. Naturally that was not at issue here, and with the data gleaned from the `.afm` files, the metrics quickly reached a usable state. Much hastened in this case since the left sidebearings are preserved in the consistent placing of the characters on the page in the source file, so only the right sidebearings needed to be checked or adjusted.

With the base character set available, next the ligatures and alternates needed to be provided for. Initially, Apple's options for Zapfino were somewhat limited as is evidenced by TextEdit.app (see Figure 3). WorldText.app, née GX/Write exposed all of the capabilities encoded within the font however, (see Figure 4) and Apple has since expanded the nsText object to fully support Zapfino's myriad of capabilities.[5] Although one may make various menu selections, or select characters from a Unicode "Character Palette" in Cocoa apps, or the Glyph palette in Adobe InDesign and other Adobe graphics applications, these options are not readily available in a text-stream composition-oriented tool like TeX.

Although OpenType and AAT are able to access characters by name, instead of directly with a number, TeX and its variants (with the exception of the new XeTeX) require a number in an encoding vector for any given character. Toward this end, an encoding scheme was worked up to allow arbitrary ligatures of up to three characters in length, and to accomodate up to 32 variants for any given (unaccented character). While that last number may seem overkill, the OpenType specification provides for up to 20 variations of a character in its `salt` tag.[6] Having 16 bits available with Omega, the available bits were split into three sets, an initial set 6 bits long and two successive sets 5 bits long. The first set is long enough to encompass basic Latin capitals and miniscule (lowercase) letters as well as the numerals 0 through 9, with two bits left over. One of these was used as a "swash" bit, while the other remains available for use. The second and third sets encompass lowercase letters.

With all characters assigned to slots it was then possible to create an Omega Translation Process (OTP) to replace characters with their appropriate ligatures:

```
input: 1;
output: 2;
states: VERBATIM;
expressions:
`0'`0' => "{\zapfinoexpert " @"035A"}";
`1'`s'`t' => "{\zapfinoexpert " @"0653"}";
...
`C'`i'`e' => "{\zapfinoexpert " @"3504"}";
`C'`o'`.' => "{\zapfinoexpert " @"35DA"}";
`D'`r'`.' => "{\zapfinoexpert " @"3A3A"}";
`E'`s'`q'`.' => "{\zapfinoexpert " @"3E50"}";
...
`g'`g' => "{\zapfinoexpert " @"ACDA"}";
...
`t'`t' => "{\zapfinoexpert " @"E27A"}";
`t'`z' => "{\zapfinoexpert " @"E33A"}";
`u'`z' => "{\zapfinoexpert " @"E73A"}";
`w'`n' => "{\zapfinoexpert " @"EDBA"}";
```

The Omega documentation[7] explains OTPs in detail. Each line in an OTP to handle a particular case is fairly straightforward. First the characters to be replaced are identified, (the `=>` indicates a replacement is being made) then a verbatim sequence of replacement commands is given within quotes, with a character's

[5] See "Panther's Major Text Services Upgrade" http://www.codepoetry.net/archives/2003/10/24/panthers_major_text_services_upgrade.php

[6] See Tag: 'salt' http://partners.adobe.com/asn/tech/type/opentype/appendices/features_pt.jsp

[7] *Draft documentation for the Omega system* (John Plaice and Yannis Haralambous, 7 March 1998 available from http://www.loria.fr/services/tex/moteurs/omega7mar1998.pdf,
*cf.* the Omega homepage at http://omega.cse.unsw.edu.au/omega/omega.jsp

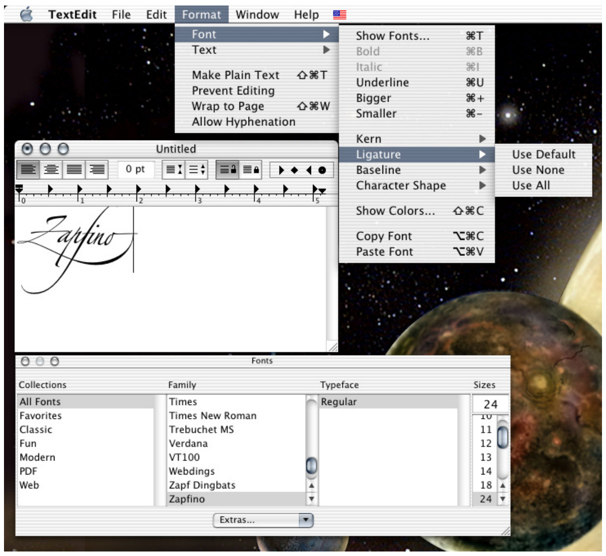
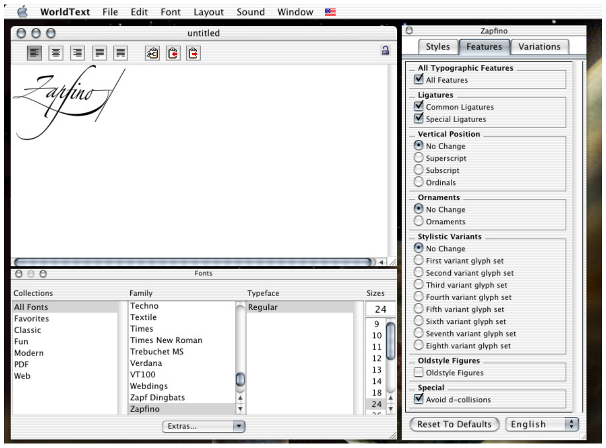


**Figure 3**: TextEdit.app options for Zapfino in Mac OS X 10.2 were rather minimalistic



All Typographic Features
[✓ All Features]

Ligatures
[✓ Common Ligatures]
[✓ Special Ligatures]

Vertical Position
[✓ No Change]
Superscript
Subscript
Ordinals

Ornaments
[✓ No Change]
Ornaments

Stylistic Variants
[✓ No Change]
First variant glyph set
Second variant glyph set
Third variant glyph set
Fourth variant glyph set
Fifth variant glyph set
Sixth variant glyph set
Seventh variant glyph set
Eighth variant glyph set

Oldstyle Figures
Oldstyle Figures

Special
[✓ Avoid d-collisions]

**Figure 4**: WorldText.app, provided in Apple's Developer Tools Samples folder provides access to all of Zapfino's capabilities

encoding being provided in an "escaped" fashion outside of the quotes (hence, `^^^^035a` is `" @"035A")`. This was saved in a file `lat2zapf.otp` and processed with the command

```
otp2ocp lat2zapf.otp lat2zapf
```

and the files moved to `\localtexmf\omega\otp` and `\localtexmf\omega\ocp` respectively.

It was then possible to test the font for the first time with a file like:

```
\font\zapfino=Zapfino at 24pt
\font\zapfinoexpert=Zapfino-expert at 24pt
\ocp\zapfOCP=lat2zapf
\ocplist\zapfOCPlist=%
\addbeforeocplist 1 \zapfOCP
\nullocplist
\nopagenumbers
\overfullrule 0pt
\zapfino\pushocplist \zapfOCPlist

00 1st 2nd 3rd 4th 5th
   6th 7th 8th 9th 10th

Cie. Co. Dr. Esq. Ht. Jr. Ltd. McSullivan
Mlle. Mme. Mr. Mrs. Ms. No. 9 Sig. Sr.
Sra. Srta. St. Thorn

dicot draw presage fez type. affianced
effect fit fluent aft egg isthmus out
Zapf phone happy Arial art mesh spell
mass stay the matte spritz uzo own

\vfill\eject\bye
```

which shows all of the ligatures, as well as points out that perhaps using old-style figures as the default wasn't such a great idea, well at least not if one intends to use the "0th" ordinal ligature.

Once the basic sidebearings of the font were set and the ligatures "wired up," the interplay of specific letterpairs needs to be addressed — the graphical interface which Mac OS X provides for Zapfino provides some hints on this, most notably, "Avoid d collisions." The standard, forward slanting lowercase *d* in Zapfino collides with the preceding character quite often, especially when preceded by a capital letter. While one may be tempted to merely set the second style of *d* described in the font file, or Linotype's Adobe Type 1 version of Zapfino as `d.2`, (Apple went this route when updating Zapfino for Mac OS X 10.3) this brings to light an excellent chance to add some "chaos" to the typeface and pick different character versions. The overall intent is to create a system which will make use of all possible characters in some circumstance, and a collection of test files which will test all such circumstances.

So instead a listing of all possible digraphs (letterpairs) in the English language was created (drawing from *Webster's Dictionary*, *The Complete Works of Shakespeare* and *The King James Bible* — NeXT users will recognize the choice of references as those bundled with NeXTstep or readily available as texts for

NeXT's Digital Librarian program), so that it was possible to determine that only *Id*, *Nd*, *Ud*, *Wd*, and *Yd* needed to be considered for the capitals, (saving by not creating entries for *Fd*, *Hd*, *Jd*, *Kd*, *Td*, *Vd*, *Xd*, and *Zd* which do not occur in the English language sample set used). The following lines in the OTP prevent such collisions by replacing the normal *d* with the `d.2` alternate.

```
    ‘I’‘d’ => "I{\zapfinoexpert " @"FC62"}";
    ‘N’‘d’ => "N{\zapfinoexpert " @"FC62"}";
    ‘U’‘d’ => "U{\zapfinoexpert " @"FC62"}";
    ‘W’‘d’ => "W{\zapfinoexpert " @"FC62"}";
    ‘Y’‘d’ => "Y{\zapfinoexpert " @"FC62"}";
```

With the obvious change taken care of, more subtle changes were then considered. Working from the most frequently occurring letters to the rarer ones, a file for each letter, containing a word for each letterpair extant in the sample texts was typeset in the font Zapfino and then examined carefully for awkward interactions. When one was found, a test file was typeset with a basic macro to set the word with all possible variations for a given letter:

```
    \def\testa#1#2#3{{#1}{#2}{#3}\par%   a
      {#1}{\zapfinoexpert ^^^^fc02}{#3}\par%
      {#1}{\zapfinoexpert ^^^^fc03}{#3}\par%
      {#1}{\zapfinoexpert ^^^^fc04}{#3}\par%
      {#1}{\zapfinoexpert ^^^^fc05}{#3}\par%
      {#1}{\zapfinoexpert ^^^^fc06}{#3}\par}
```

allowing the selection of the best choice. The pair *nf* and a number of other pairs ending in *f* did not work well because the *f* in Zapfino One does not connect to the character before it, so a number of lines similar to the following were added to the OTP to handle such cases:

```
    ‘n’‘f’ => "n{\zapfinoexpert " @"FCA4"}";
```

Examining that file will show the reader which character pairs were felt to require adjustment. For the most part, replacing one character or the other (occasionally both) with an alternate provided an æsthetically pleasing combination, though a few uncommon pairs (*gj, Kz*) did not yield such.

Zapfino's many lowercase ligatures required that the list of digraphs be extended to encompass the trigraphs and tetragraphs which encompass these lettersets in addition to the digraphs mentioned above. For each such letter or ligature there is a file, (e.g., `di-.txt`) which encompasses the extant lettersets including said letter or ligature's letters with the hyphen indicating the location of the letters being checked for, so that file ranges from "Diagram" through "dizzy." Since the *di* and *dr* ligatures do not reach as far to the left, most such letterpairs were acceptable. A notable exception is "ldr" and "ldi" which were set to make use of the alternate character `d.4`.

Zapfino is so narrow, however, with such a low x-height, and such large ascenders, that it is actually possible for the ascender on a *d* to collide with a character two characters before it as in, "led." Thus a set of test files encompassing all characters with ascenders followed by any letter, followed by a "d" was created. Unfortunately, it is also possible for a "d" at the beginning of a word to clash with the previous word if it ends with a descender. Suggestions for dealing with this situation would be welcome.

Doubled characters require special attention if there is no ligature for them, requiring that one check to determine which letter versions look attractive together.

After all such files were typeset and examined twice (once to establish the initial replacements, a second time to check for interactions between the replacements and characters which follow or precede them) it was possible to begin using Zapfino in Omega.

### Swash markup and packaging

I'm planning on having the setup be as straightforward as possible — hopefully I can manage to create a LaTeX package so that it'll be just a matter of

```
\usepackage{omegazapfino}
...
\ZapfinoText{This would be typeset in Zapfino w/ contextual ligatures.}
\ZapfinoRebus{This would have ornaments/pictures for words like (-(duck)-) and (-(pen)-)}
```

I still need to work up a markup scheme for accessing arbitrary alternates and swashes though.

I've pretty much decided on the paren hyphen paren scheme for marking rebus text. That way one could turn it off for proofing by searching / replacing `(-(` w/ `(~(`, though I guess w/ a LaTeX package the correct thing to do would be to have it honour the "draft" option in the documentclass if present and set text then.

For swashes I was thinking something along the lines of:

| | |
|---|---|
| `----a` | (that'd be the first level) |
| `+----a` | (second) |
| `++----a` | (third) |
| `+++----a` | (fourth) |
| `\----a` | (an upward swash) |
| `/----a` | (a downward swash) |

and reversing everything for forms appropriate to the ends of words — does that sound okay?

That way I could run checks (moving the em- and en-dashes to the end of the OTP) to remove them when they were added for letterforms which lacked such. Now that still leaves the matter of alternates in the middle of words... perhaps the paren–hyphen–paren option would work for that? Hmm, `co(-(g)-)ent` — is that too ugly?

```
co(--(g)--)ent
co(---(g)---)ent
```

That's not so bad, is it? Then again, Zapfino has up to *eight* variations for some letters... so counting all those hyphens might get kind of tedious.

## Peace

The first such use was to set a holiday card modelled on Jeanyee Wong's famous polyglot card for UNICEF.[8] After this was initially set, it was announced on Usenet and various mailing lists related to TeX or typography. Jef Tombeur and Apostolos Syropoulos were kind enough to provide translations for French and Greek respectively (though since Zapfino doesn't contain a full Greek alphabet the latter was provided typeset in the Kerkis font), and a revised version was made available as a part of the TEX Showcase.[9] In preparation for this paper, a second announcement and request for translations was made, with many people providing translations and commentary (I had neglected to mention the text, "Peace on earth, good will toward men" as being the latter part of the verse Luke 2:14 from *The King James Bible*, so in addition to straightforward transcriptions from various Bibles, I also received a number of personal interpretations) resulting in an expansion of the card to three pages, almost a dozen languages (Greek has been temporarily ommitted) and twenty names. Once typeset as a three page `.pdf` the first and last pages which were not filled completely were cropped, then each page was exported as a `.eps` and imposed in Macromedia FreeHand where each was scaled appropriately to fit nicely on a tabloid or A3 sheet which could then be folded twice to make a gatefold card. This affords the illusion of having three different sizes of Zapfino available, despite only one font at a fixed size (24 pt.) having been made.

## And Pictures Too

In addition to alternates and ligatures, Zapfino also has a number of ornaments available. Oddly, although Helvetica seems to have a "Rebus" option in Mac OS X 10.3 "Panther," this does not seem to have been implemented for Zapfino, reducing one to point-and-click to access them from a glyph palette of some sort.

[8] Exhibit 214. UNICEF Christmas Card. Jeanyee Wong. 1962. *Two Thousand Years of Calligraphy: A Three-part exhibition organized by the Baltimore Museum of Art, the Peabody Institute Library and the Walters Art Gallery, June 6–July 18, 1965, A Comprehensive Catalog*: Baltimore, Maryland, 1965.

[9] http://www.tug.org/texshowcase

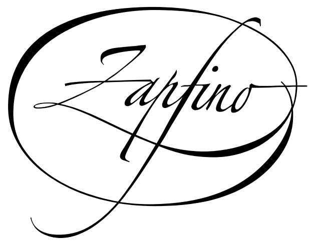


**Figure 5**: Zapfino ligature ornament

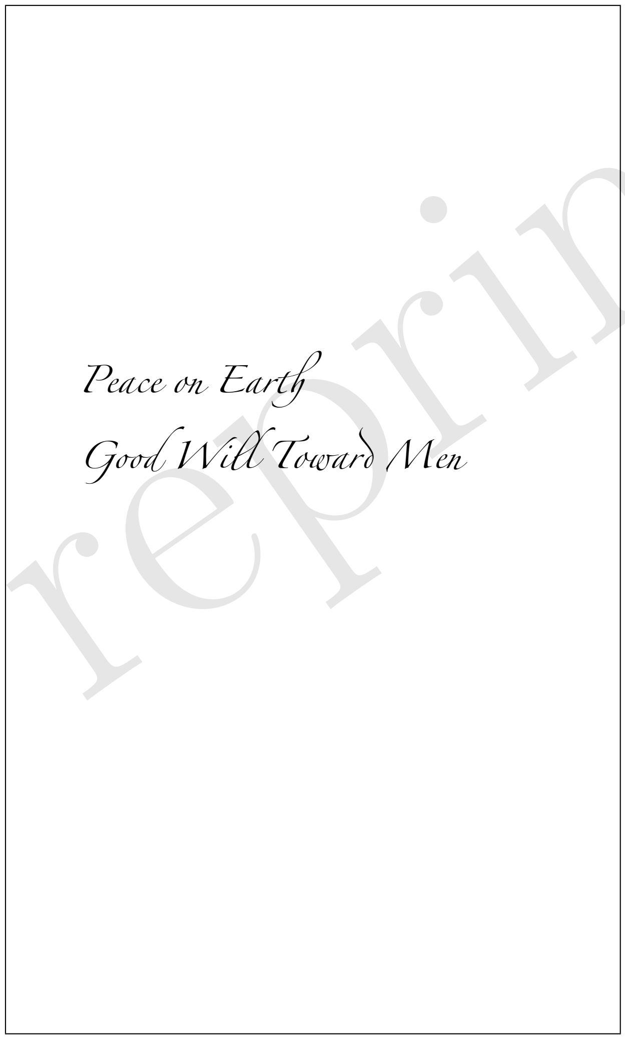
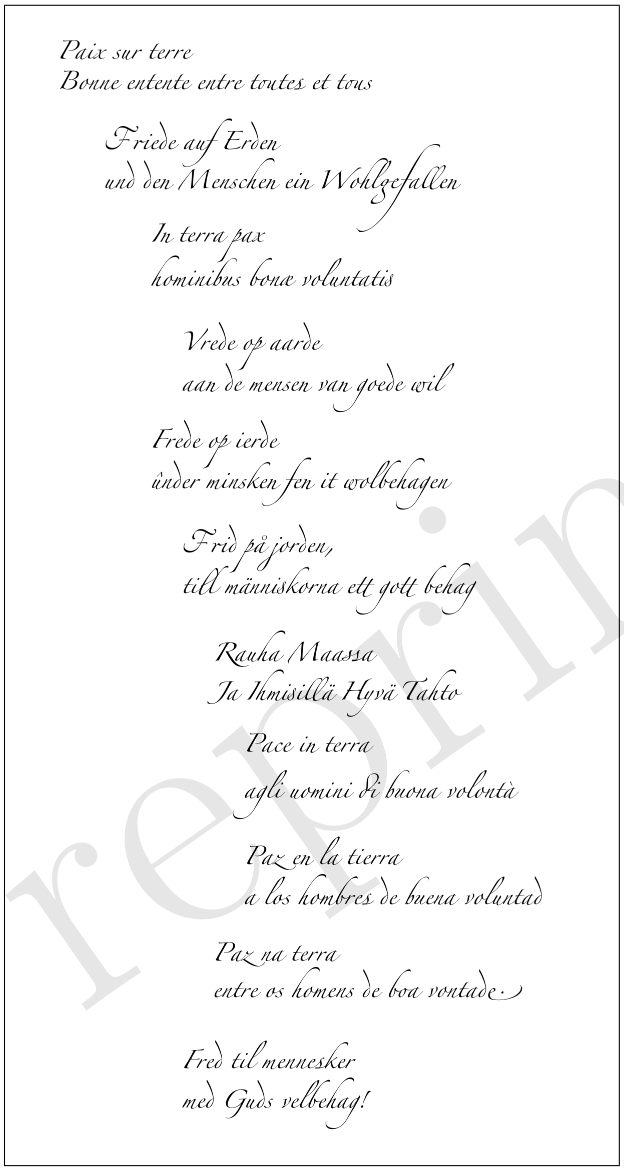


**Figure 6**: Peace on Earth card front page



Paix sur terre
Bonne entente entre toutes et tous

Friede auf Erden
und den Menschen ein Wohlgefallen

In terra pax
hominibus bonæ voluntatis

Vrede op aarde
aan de mensen van goede wil

Frede op ierde
ûnder minsken fen it wolbehagen

Frid på jorden,
till människorna ett gott behag

Rauha Maassa
Ja Ihmisillä Hyvä Tahto

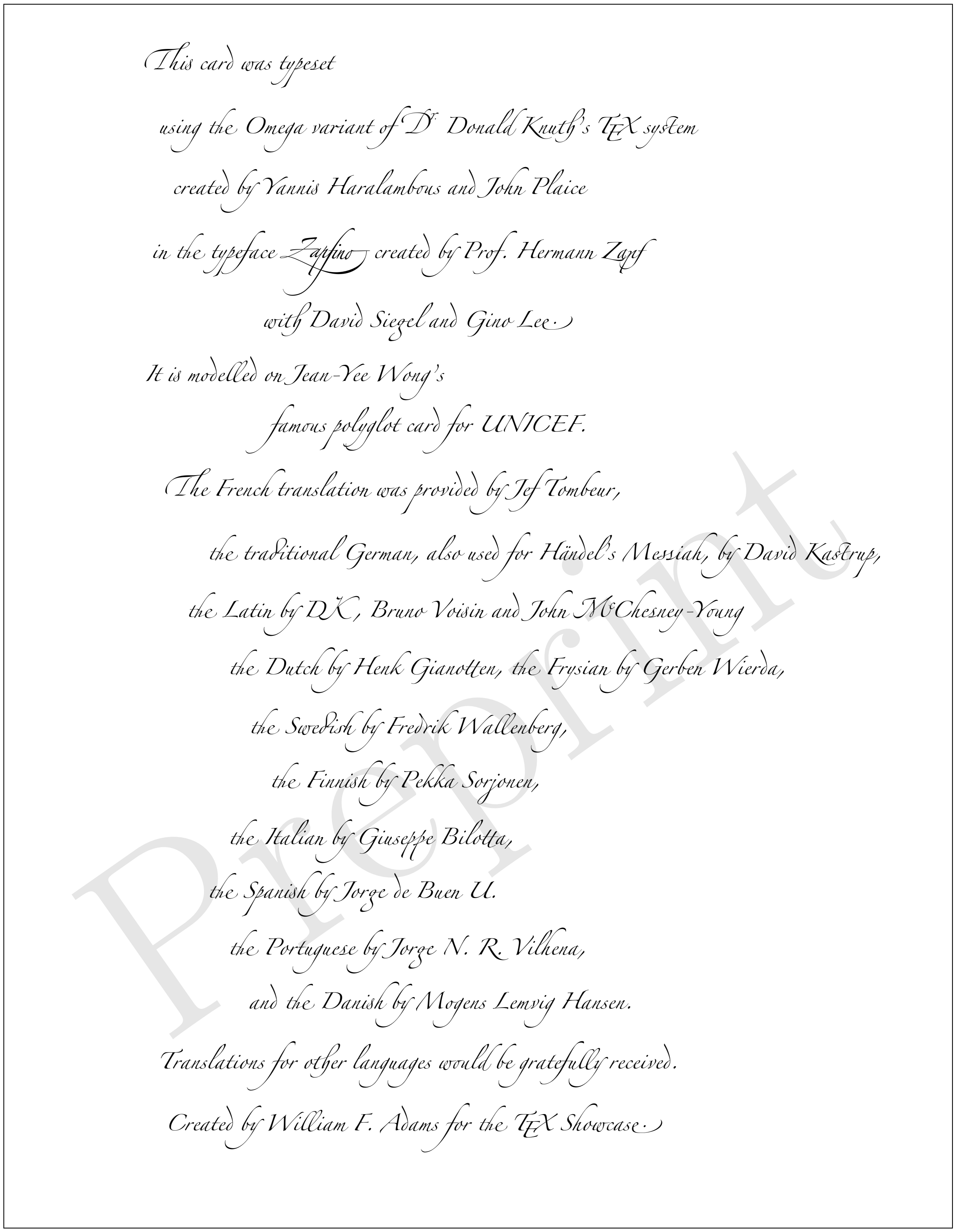
Pace in terra
agli uomini di buona volontà

Paz en la tierra
a los hombres de buena voluntad

Paz na terra
entre os homens de boa vontade

Fred til mennesker
med Guds velbehag!

**Figure 7**: Peace on Earth interior



This card was typeset

using the Omega variant of Dr Donald Knuth's TeX system

created by Yannis Haralambous and John Plaice

in the typeface Zapfino created by Prof. Hermann Zapf

with David Siegel and Gino Lee.

It is modelled on Jean-Yee Wong's

famous polyglot card for UNICEF.

The French translation was provided by Jef Tombeur,

the traditional German, also used for Händel's Messiah, by David Kastrup,

the Latin by DK, Bruno Voisin and John McChesney-Young

the Dutch by Henk Gianotten, the Frysian by Gerben Wierda,

the Swedish by Fredrik Wallenberg,

the Finnish by Pekka Sorjonen,

the Italian by Giuseppe Bilotta,

the Spanish by Jorge de Buen U.

the Portuguese by Jorge N. R. Vilhena,

and the Danish by Mogens Lemvig Hansen.

Translations for other languages would be gratefully received.

Created by William F. Adams for the TeX Showcase.

**Figure 8**: Peace on Earth card back page, greatly enlarged