**At the deepest level, we could also fiddle with the subroutine definitions in *cmbase.mf* – and of course that would essentially amount to the creation of a new family of fonts.**

— DONALD E. KNUTH, *The Computer Modern Family of Typefaces* (1980)

**. . .All hell broke loose.**

— JOHN MILTON, *Paradise Lost, i, 917* (1667)

# The OCLC Roman Family of Fonts

Georgia K.M. Tobin
OCLC Online Computer Library Center, Inc.

"A complete font design is a complex system . . . ." As near as I can tell, everything in Don Knuth's marvelous book, *The Computer Modern Family of Typefaces*, is true; but there is no one statement whose truth I will so readily avow as the preceding sentence. My experience with the METAFONT design of OCLC ROMAN has shown me that a thorough understanding of what is entailed in the complex system that is called a family of fonts is essential to successful METAFONT design. In this report, I will try to show how the complexities of the OCLC ROMAN family of fonts have been incorporated into METAFONT file structure, and to describe the files which fit into that file structure. My intention is to show how one would create a family of fonts and, by implication at least, to offer supporting evidence for the power and subtlety of METAFONT as a graphic design tool.

The creation of workable versions of the fonts outlined was accomplished between March and October of 1983 by a font design group consisting of myself; the fine-tuning of these workable versions is an ongoing process. I had the pleasure of briefly discussing my designs with Dr. Knuth, but the basis of my design technique was gleaned from perusal of *The Computer Modern Family of Typefaces*, and application of the ideas contained therein to the peculiarities of OCLC ROMAN. That, combined with experience gained from my earlier METAFONT design work, combined with an inborn desire to run a tidy operation, gave rise to the OCLC ROMAN file structure as it now stands.

My OCLC ROMAN family of fonts is intended to capture the flavor of Stanley Morison's *Times Roman* in the METAFONT idiom. *Times Roman* is one of the most popular, most readable, and most read typefaces ever designed; it is also, in my opinion, one of the most beautiful. The OCLC ROMAN family of fonts includes such standard *Times Roman* fonts as text, italic, bold, titling, and extended titling. It also includes

a complete symbols font, an italic font suitable for typesetting mathematics, and an extensible font (i.e., one whose characters can grow as required by the formula being set) consistent with Dr. Knuth's Computer Modern fonts. Furthermore, there are fonts suitable for typesetting text in Cyrillic or in Greek.

The first point I want to stress is the breadth of the concept of a METAFONT family of fonts, a system about which the following statements are true:

> The fonts which make up a family of fonts are clearly distinguishable from one another, yet all share certain common traits which identify them as members of that family of fonts.
>
> The characters which make up a font are clearly distinguishable from one another, yet all share certain common traits which identify them as members of that font.
>
> The structural components which make up the characters in a font are clearly distinguishable from one another, yet all share certain common traits which permit them to be integrated into a distinguishable character of a distinguishable font of a distinguishable family of fonts.

The single, particular character thus created must be capable of being simultaneously the *specific* representation of a particular character in a particular font of a particular family of fonts, and a *general* template for that particular character at any point size and at any resolution.

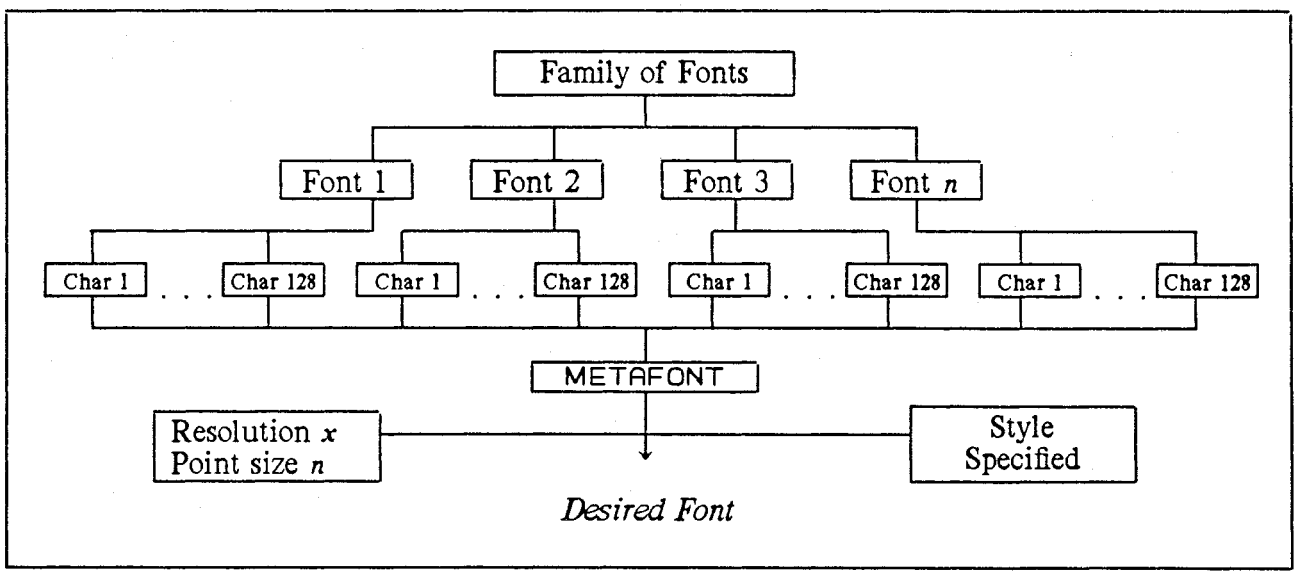


**Figure 1. The Hierarchy in a Family of Fonts**

My experience with OCLC ROMAN has indicated that this rather tall order may be filled by:

1. Attention to the hierarchical relationship of characters to fonts to a family of fonts in the METAFONT file structure;
2. Attention to the niceties of the particular level in the hierarchy with which a file is concerned; that is, a "font" level file should take care of "font" level detail, not "family" level detail or "character" level detail.

The file structure of OCLC ROMAN corresponds to this hierarchy. *Base.mf* takes care of details at the "family of fonts" level. It contains the definitions and subroutines used by each and every character in each and every font of the OCLC ROMAN family; it also contains subroutines available to each and every character, though these subroutines may not be needed to depict a given form.

Four files take care of details at the "font" level. The first of these is the name of the style of OCLC ROMAN to be constructed, e.g. *text.mf*, *italic.mf*, *bold.mf*, etc. This file assigns values to a number of variables, including: the heights of upper- and lower-case letters; the various pens used in drawing the font; the amount of intercharacter spacing in the font; and the amount of slant in the font. This *font.mf* file calls the appropriate *font_setwidths.mf* file, which allows the designer to assign values for the width of the main body of the character and of the right and left sidebearings for each character in a particular font. *Font_ligatures.mf* provides METAFONT with some critical information on the proper kerning of the font and what ligatures (if any) are used. Equally important, it gives the font's TEXinfo. This consists of at least seven parameters which control such critical items as interword spacing, shrink and stretch, and slant per point. The fourth file, *font_switch.mf*, merely specifies the characters that will make up the font.

At the "character" level of detail, we have the character routines for the font. Each of these is a short program (usually twenty to fifty lines of METAFONT code) which describes a particular character in a general way; that is, it describes the character in terms such that METAFONT may draw it at any point size and for any resolution.

We need to consider with some care how these files interact. I shall attempt to do this by starting at the "top," at the family of fonts level of detail, and working my way downward to the individual character level, showing where we obtain the information we require along the way. Let us therefore take a long, hard look at *base.mf*.

The first small bit of *base.mf* contains the METAFONT code which takes care of various housekeeping tasks. These values hold for the entire OCLC ROMAN family. *Mode* is a value assigned by the designer, which must be shared by each font in a family of fonts, because it sets up the values of two factors used throughout METAFONT's computations: *pixelshoriz* and *pixelsvert*. These represent, respectively, the number of pixels oriented horizontally and the number of pixels oriented vertically, and they control the resolution of the output which is produced. The next big chunk of code in *base.mf* is a subroutine called *romanbegin.mf*. This code pertains to the *font* level. It provides METAFONT with such crucial information as what point size of font is to be designed, what sort of grid the font will be designed upon, and what sorts of pens will be used. My insistence upon "sorts of" in the final two clauses of the preceding sentence is not some stylistic penchant for vagueness, but an accurate representation of what *base.mf* does. For though, as I mentioned, *base.mf* is the largest and most complex OCLC ROMAN file read by METAFONT, without additional instructions from the designer it will draw absolutely nothing. (Its saving grace, of course, is that, with the proper additional instructions, it is quite a prolific draftsman.)

As an example, let us consider the way in which the grid upon which OCLC ROMAN characters are designed is defined by *base.mf*. The code for the two routines that accomplish this is shown in Figure 2. We draw six horizontal lines of uniform length at 1) the lowest point which characters with non-rounded descenders reach ($-d$); 2) the baseline upon which all characters sit (0); 3) the greatest height which lower-case characters without ascenders reach ($m$); 4) the greatest height which nonrounded upper-case letters reach ($h$); 5) the greatest height which rounded upper-case letters reach (*topp*, i.e. $h + vo$); and 6) the lowest point which characters with rounded descenders reach (*bott*, i.e. $-d - vo$). Only the $y$ value of the baseline is nonnegotiable; all the other $y$ values mentioned in the preceding sentence vary from font to font, and are passed to this routine from whichever *font.mf* file the designer has specified. Each of these six lines starts at $x = 0$ and ends at $x = r$, where $r$ is the width of an individual character which varies from character to character and is passed to this routine from whichever *font_setwidths.mf* file the designer has specified. The grid is completed by drawing two vertical lines from the highest point to the lowest point, one at $x = 0$ and one at $x = r$.

The important point to notice here is not the mechanics of constructing the OCLC ROMAN grid, but the way in which a METAFONT subroutine provides for both the underlying coherence of a family of fonts and the unique qualities of a particular font and character. That is, the grid always extends from $h + vo$ to $-d - vo$, but the particular values of $h$, $d$, and $vo$ vary according to design considerations.

Thus far, we have discussed that portion of *base.mf* which pertains to each and every character in a given font. All of the remaining subroutines in *base.mf* provide rote ways to draw certain common

features of individual characters. These move down a level in the hierarchy from the first part of *base.mf*, but provide the same flexibility at that level. That is, they allow for both the individuality of a font and the similarity of a family of fonts.

```
subroutine box:
new offset; offset= 0;
no drawtrace; no proofmode;
new topp, bott, leftt, rightt;
topp=h+vo; bott=-d-vo;
x1 = x3 = x5 = x7 = x9 = x11 = x13 =leftt;
x2 = x4 = x6 = x8 = x10 = x12 = x14 =rightt;
y1 = y2 =-d;
cpen;
1 draw 1 . . 2
y3 = y4 =0;
draw 3 . . 4;
y5 = y6 = m;
draw 5 . . 6;
y7 = y8 = h;
draw 7 . . 8;
y9 = y10 = topp;
draw 9 . . 10;
y13 = y14 = bott;
draw 13 . . 14;
trxy 0;
if italcorr> 0:
x19 = x20 =rightt+ italcorr·pixelshoriz;
y19 =topp;  y20 = 0;
fi;
trxy pixelsvert·pixelshoriz·slant;
call unitlines

subroutine unitlines:
y1 =topp;  y2 =bott;  cpen;
new x1, x2;  x1 = x2 = 0;  draw 1 . . 2;  % left
new x1, x2;  x1 = x2 = r;  draw 1 . . 2;  %right
```

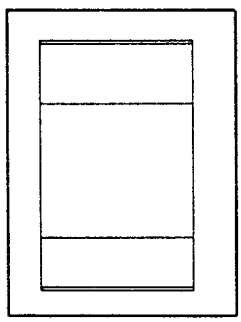

**Figure 2. "Font Level" subroutine from *base.mf* and the grid it draws**

Let me backtrack for just a bit to enlarge on what I mean. One of the first things that we notice when contemplating the design of a Times Roman-like font is that we will need to draw rather a lot of serifs. In fact, we will need (among others) strictly *horizontally* oriented serifs which extend to either the right or the left of the letter's stem, or to *both* the right and the left of the stem, and strictly *vertically* oriented serifs, which extend either upwards or downwards from the bar, or both upwards and downwards from the bar. We will also need several different sorts of sloped serifs. By describing ways to draw these various serifs in subroutines in *base.mf*, we have a way to both preserve the inherent "Times Roman-ness" of each and to allow a particular serif to look right for a given style of a given letter at a given point size. To better understand how this is done, let us consider *symmhserif*, the subroutine for a horizontally oriented serif which extends to both the right and the left of the letter's stem at either the top or the bottom of a stem.

Whatever routine calls *symmhserif* must pass it four arguments. *Midedge* is the point exactly halfway between the leftmost and rightmost points on the serif at its topmost (for a top-of-stem serif) or bottommost (for a bottom-of-stem serif) point. *Joinstem* is the point in the middle of the stem as high as the point at which the serif joins the stem. If $y_{joinstem}$ is less than $y_{midedge}$, *symmhserif* knows that it is dealing with a top-of-stem serif; if $y_{joinstem}$ is greater than $y_{midege}$, it is dealing with a bottom-of-stem serif. *Stempen* is the name of the horizontal pen with which the stem to which this serif is to be connected is drawn. *Serifwd* is the width of the serif.

Fortified with this knowledge, *symmhserif*'s plodding brain proceeds in the following manner: (See Figure 3 to follow along.)

> I will draw the portion of the serif which extends to the left of the stem first. I will do that in the following way. I will define a point 1 on the leftmost edge of the stem and as high as the point *midedge*. I will define a point 2 on the leftmost edge of the stem and as high as the point *joinstem*. I will define a point 3 which lies to the left of the middle of the stem by a distance equal to one half the total width of the serif and as high as the point *midedge*. I will define a point 4 which is the same distance to the left of the middle of the stem as 3 but which is either a minimal pen height *above* the point *midedge* (if I'm drawing a bottom-of-stem serif) or a minimal pen height *below* the point *midedge* (if I'm drawing a top-of-stem serif). I will define a point 5 which lies one half of the way from point 1 to point 3 and is as high as point 3. I will define a point 6 which lies on a concave curve between points 4 and 2; the arc of this curve I know from the value of *brangle*, which I got from this particular font's *font.mf* file.
>
> Now, I am ready to draw. I will use a circular pen one pixel in diameter. I fill in the entire area bounded by the curve from 4 to 2 and the segment from 3 to 1. That takes care of the serif to the left side of the stem.
>
> Now, I use a horizontal pen of size *stempen* to extend the stem to the bottom of the serif.
>
> Now, I am ready to draw the portion of the serif which extends to the right of the stem. I will do that in the following way. I will define a point 7 on the rightmost edge of the stem and as high as the point *midedge*. I will define a point 8 on the rightmost edge of the stem and as high as the point *joinstem*. I will define a point 9 which lies to the right of the middle of the stem by a distance equal to one half the total width of the serif and as high as the point *midedge*. I will define a point 10 which is the same distance to the right of the middle of the stem as 9 but which is either a minimal pen height *above* the point *midedge* (if I'm drawing a bottom-of-stem serif) or a minimal pen height *below* the point *midedge* (if I'm drawing a top-of-stem serif). I will define a point 11 which lies one half of the way from point 7 to point 9 and is as high as point 9. I will define a point 12 which lies on a concave curve between points 10 and 8; the arc of this curve I know from the value of *brangle*, which I got from this particular font's *font.mf* file.
>
> Now, I am ready to draw that side of the serif. I will use a circular pen one pixel in diameter. I fill in the entire area bounded by the curve from 10 to 8 and the segment from 9 to 7. That takes care of the serif to the right side of the stem, and I'm all done.

It is clear enough that having the subroutine *symmhserif* will spare us the task of cranking out all that code every time we want a serif; but it does more, too. Values which are defined in a font's *style.mf* file are used in *symmhserif*, both explicitly (e.g. *brangle*) and implicitly (e.g. *serifwd* is calculated from *standardserif*). This tends to make any serif look as though it were drawn by the same hand; and that lends underlying stylistic coherence to the entire family of fonts. That coherence is really the whole point of *base*.

```
subroutine symmhserif (index midedge)
                      (index joinstem)
                      (index stempen)
                      (var serifwd)
% This draws a symmetrical horizontal serif
cpen;
no proofmode;
new w99;   w99 = (wstempen);
call checkpen(99);
rt99 x1 =good99(xmidedge);
rt99 x2 =good99(xjoinstem);
y1 = ymidedge; y2 = yjoinstem;
x3 = x4 = xmidedge - (.5 * r*serifwd);
y3 = ymidedge;
if ymidedge > yjoinstem:
top101 y4 = y3;
else:
bot101 y4 = y3;
fi;
x5 = ½[x3, x1];
y5 = y3;
x6 = 1/brangle[x4, x2];
y6 = 1/brangle[y2, y4];
cpen;
1 ddraw 3 . . 5 . . 1,
4{x2 - x4, 0}  . . 6{x2 - x4, y2 - y4}  . . 2{0, y2 - y4} ;
hpen;  hpenht 1;
wstempen draw midedge. . joinstem;
lft99 x7 =good99(xmidedge);
y7 = ymidedge;
lft99 x8 =good99(xjoinstem);
y8 = yjoinstem;
x9 = x10 = xmidedge + (.5 * r*serifwd);
y9 = ymidedge;
y10 = y4;
y11 = ½[x9, x7];
y11 = y9;
x12 = 1/brangle[x10, x8]; y12 = 1/brangle[y8, y10];
cpen; 1 ddraw 9 . . 11 . . 7,
10{x8 - x10, 0}  . . 12{x8 - x10, y8 - y10}  . . 8{0, y8 - y10} ;
```

**Figure 3. A "Character Level" subroutine from *base.mf***

The deepest level in our family of fonts hierarchy is the character level. At this level, the designer uses information from the other files at higher levels described above to compose a description of a character which at once distinguishes it from other characters in the font, enables it to fit in stylistically with other characters in the font, and, in true METAFONT form, makes it adaptable to other point sizes and resolutions.

```
"The Letter I"
call charbegin(73,ph,0,ph*slant);
cpen;
new serif; serif=standardserif/charw;
x1 = x2 = x3 = x4 =good7(r/2);
y1 =h; y2 =h-serifht;
y3 = 0; y4 = 0+serifht;
call symmhserif(1,2,7,serif); call symmhserif(3,4,7,serif);
hpen; hpenht 1; w7 draw 2 . . 4.
```

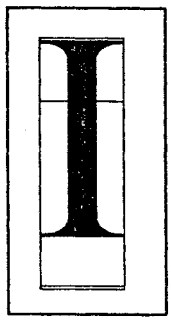

**Figure 4. A Character Subroutine, and the Character It Draws.**

Consider the routine for a capital "I". This is quite a simple routine, and yet even here, there is a lot going on. *Standardserif* is a value defined in each font's style.mf file; *charw* is a value calculated in *base.mf* using information culled from the *style_ligature* file and the *style.mf* file. By setting the quotient of these two values equal to the width of the serif passed to *symmhserif* not only in the routine for the letter I but for all letters, the designer is assured that the width of the serifs will be uniform throughout the alphabet. Likewise, the pen $w_7$ is defined in *base.mf* using values obtained from the *style.mf* file; and we know with confidence that the pen will be the same for all letters that share the same definition of $w_7$, that is, all letters in a given font.

The upshot of all this interdependence among the various files which make up the font family OCLC ROMAN is this: by carefully designing a basic collection of character forms and carefully setting up the subroutines which support those letter forms, we can exploit the inherent flexibility of our system to produce a limitless number of variations on the typographic theme we have chosen to produce a rich and varied but stylistically coherent family of fonts through skillful manipulation of our files. All of which sounds stirring enough; but we must eschew such generalities for a good long look at how this translates into the nitty-gritty of real fonts.

We first produce a standard text font:

| | *'0* | *'1* | *'2* | *'3* | *'4* | *'5* | *'6* | *'7* | |
|---|---|---|---|---|---|---|---|---|---|
| *'00x* | Γ | Δ | Θ | Λ | Ξ | Π | Σ | Υ | "0x |
| *'01x* | Φ | Ψ | Ω | ff | fi | fl | ffi | ffl | |
| *'02x* | ı | ȷ | ` | ´ | ˇ | ˉ | ˘ | ° | "1x |
| *'03x* | ¸ | ß | æ | œ | ø | Æ | Œ | Ø | |
| *'04x* | ‘ | ! | ” | # | $ | % | & | ’ | "2x |
| *'05x* | ( | ) | * | + | , | - | . | / | |
| *'06x* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | "3x |
| *'07x* | 8 | 9 | : | ; | i | = | ¿ | ? | |
| *'10x* | @ | A | B | C | D | E | F | G | "4x |
| *'11x* | H | I | J | K | L | M | N | O | |
| *'12x* | P | Q | R | S | T | U | V | W | "5x |
| *'13x* | X | Y | Z | [ | “ | ] | ^ | ˙ | |
| *'14x* | ‘ | a | b | c | d | e | f | g | "6x |
| *'15x* | h | i | j | k | l | m | n | o | |
| *'16x* | p | q | r | s | t | u | v | w | "7x |
| *'17x* | x | y | z | – | — | ˝ | ~ | ¨ | |
| | "8 | "9 | "A | "B | "C | "D | "E | "F | |

**Figure 5. OCLC ROMAN text 10 point.**

By establishing another *style.mf* file which sets *slant* to 0.20 rather than 0, but using those same characters, we get:

| | '0 | '1 | '2 | '3 | '4 | '5 | '6 | '7 | |
|---|---|---|---|---|---|---|---|---|---|
| '00x | Γ | Δ | Θ | Λ | Ξ | Π | Σ | Υ | "0x |
| '01x | Φ | Ψ | Ω | ff | fi | fl | ffi | ffl | |
| '02x | ı | ȷ | ` | ´ | ˇ | ˘ | ¯ | ° | "1x |
| '03x | ¸ | ß | æ | œ | ø | Æ | Œ | Ø | |
| '04x | ˝ | ! | ” | # | $ | % | & | ’ | "2x |
| '05x | ( | ) | * | + | , | - | . | / | |
| '06x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | "3x |
| '07x | 8 | 9 | : | ; | i | = | ¿ | ? | |
| '10x | @ | A | B | C | D | E | F | G | "4x |
| '11x | H | I | J | K | L | M | N | O | |
| '12x | P | Q | R | S | T | U | V | W | "5x |
| '13x | X | Y | Z | [ | “ | ] | ^ | ˙ | |
| '14x | ‘ | a | b | c | d | e | f | g | "6x |
| '15x | h | i | j | k | l | m | n | o | |
| '16x | p | q | r | s | t | u | v | w | "7x |
| '17x | x | y | z | – | — | ˝ | ~ | ¨ | |
| | "8 | "9 | "A | "B | "C | "D | "E | "F | |

**Figure 6. OCLC ROMAN slanted text 10 point.**

We need to do a bit more work to obtain a true italic, which looks like this:

| | '0 | '1 | '2 | '3 | '4 | '5 | '6 | '7 | |
|---|---|---|---|---|---|---|---|---|---|
| '00x | Γ | Δ | Θ | Λ | Ξ | Π | Σ | Υ | "0x |
| '01x | Φ | Ψ | Ω | ff | fi | fl | ffi | ffl | |
| '02x | ı | ȷ | ` | ´ | ˇ | ˘ | ¯ | ° | "1x |
| '03x | ¸ | ß | æ | œ | ø | Æ | Œ | Ø | |
| '04x | ˝ | ! | ” | # | $ | % | & | ’ | "2x |
| '05x | ( | ) | * | + | , | - | . | / | |
| '06x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | "3x |
| '07x | 8 | 9 | : | ; | i | = | ¿ | ? | |
| '10x | @ | A | B | C | D | E | F | G | "4x |
| '11x | H | I | J | K | L | M | N | O | |
| '12x | P | Q | R | S | T | U | V | W | "5x |
| '13x | X | Y | Z | [ | “ | ] | ^ | ˙ | |
| '14x | ‘ | a | b | c | d | e | f | g | "6x |
| '15x | h | i | j | k | l | m | n | o | |
| '16x | p | q | r | s | t | u | v | w | "7x |
| '17x | x | y | z | – | — | ˝ | ~ | ¨ | |
| | "8 | "9 | "A | "B | "C | "D | "E | "F | |

**Figure 7. OCLC ROMAN Italic 10 point.**

All we needed to do this was to create a *style.mf* file which establishes a *slant* value of 0.20 and defines a set of slightly thinner pens than those used for text. We were able to use the same character forms by and large, except for the lower case letters. We also needed to make some changes in the character widths. However, the bit of extra work pays off; for by running these character routines with a *style.mf* file identical to the one used for italic except that it sets *slant* to 0, we get an unslanted italic font.

Of course, not all the various fonts we want are produced with so few changes from standard text. **Boldface**, for example is *not* simply text written with broader pens; there are fundamental differences in several of the character forms, and the proportions in bold lettering are *not* merely the proportions of text scaled up. Happily, though, our file structure enables us to describe all these differences in the *style.mf*, *style_ligatures.mf*, and *style_setwidths.mf* files, and to use the subroutines in *base.mf*.

This same state of affairs is what helps to make some fonts, which require almost a complete new set of character forms, to nevertheless have an OCLC ROMAN look to them. Consider, for example, the following:

| | *'0* | *'1* | *'2* | *'3* | *'4* | *'5* | *'6* | *'7* | |
|---|---|---|---|---|---|---|---|---|---|
| *'00x* | Г | Δ | Θ | Λ | Ξ | Π | Σ | Τ | "0x |
| *'01x* | Φ | Ψ | Ω | ю | ж | ь | ъ | э | |
| *'02x* | № | « | ` | ´ | ˘ | ¯ | ¨ | ˙ | "1x |
| *'03x* | ¸ | » | и | Ю | Ж | Ь | Ъ | Э | |
| *'04x* | ˛ | ! | ” | # | $ | % | & | ’ | "2x |
| *'05x* | ( | ) | * | + | , | - | . | / | |
| *'06x* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | "3x |
| *'07x* | 8 | 9 | : | ; | ¡ | = | ¿ | ? | |
| *'10x* | Й | А | Б | Ц | Д | Е | Ф | Г | "4x |
| *'11x* | Х | И | Я | К | Л | М | Н | О | |
| *'12x* | П | Ч | Р | С | Т | У | В | Щ | "5x |
| *'13x* | Ш | Ы | З | [ | “ | ] | ^ | · | |
| *'14x* | ‘ | а | б | ц | д | е | ф | г | "6x |
| *'15x* | х | и | я | к | л | м | н | о | |
| *'16x* | п | ч | р | с | т | у | в | щ | "7x |
| *'17x* | ш | ы | з | – | — | ” | ~ | ¨ | |
| | "8 | "9 | "A | "B | "C | "D | "E | "F | |

**Figure 8. OCLC ROMAN compatible Cyrillic 10 point.**

Despite their superficial dissimilarity, 'I' and 'И' are more nearly akin than 'I' and 'I'. (The second *I* in the preceding sentence is from Dr. Knuth's Computer Modern Roman alphabet.) The first pair share pens; they share serif routines; they share *h*-heights. The second pair share none of those things, and strike us as similar because they are, coincidentally, representations of the same English character.

The font family resemblance becomes more difficult to descry in non-text fonts like the math symbols font or the extensible symbols font.

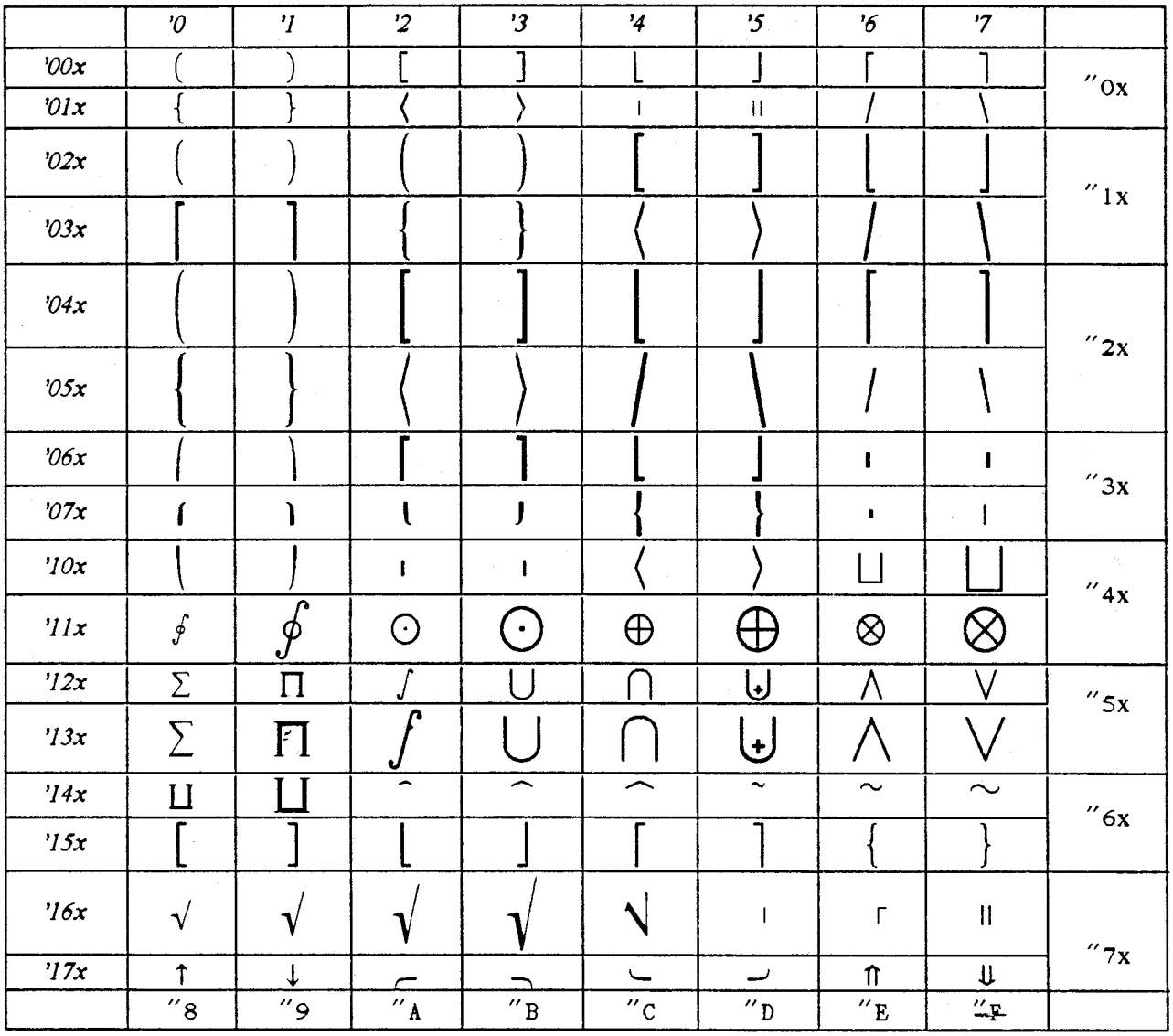


**Figure 9. OCLC ROMAN compatible Extensible Symbols 10 point.**

Nevertheless, it is underlying family resemblance among the 600-odd routines which compose the OCLC ROMAN family which enable apparently dissimilar fonts to work together harmoniously. Witness the following:

OCLC ROMAN has a font which corresponds to every font in the file `plain.tex` used by TEX82. In fact, we at OCLC are using a version of `plain.tex` with the font families redefined to OCLC ROMAN .

The OCLC ROMAN family of fonts encompasses a complete range of digital typefaces suitable for a broad spectrum of uses from regular text to scientific and mathematical material.

Text
*Italic*
**Medium Bold**
***Medium Bold Italic***
**Bold**
***Bold Italic***
**Extra Bold**
***Extra Bold Italic***
TITLING
**HEAVY TITLING**
**EXTENDED TITLING**
*Slanted Text*
Unslanted Italic
Monospace

The complete Roman alphabet is available in text, medium bold, bold, and extra bold weights, with corresponding italics. Roman small caps ("titling" faces) are available in text and bold weights, as well as in an extended version. Specialty versions of the Roman alphabet include slanted text, unslanted italic and a monospaced face.

Пролетарии всех стран соединяйтесь!

Non-Roman alphabets include Greek, Cyrillic for modern Russian, Azeri, and a "Slavic" character set containing the letters used in Bulgarian, Byelorussian, Macedonian, Serbian and Ukrainian; the latter two are available in text weight in both standard and slanted versions.

*The sum of* $\left\{\begin{matrix} 1+2 \\ 2+2 \\ 3+2 \end{matrix}\right\}$ *is* $\left\{\begin{matrix} 3 \\ 4 \\ 5 \end{matrix}\right\}$.

$$\pi(n) = \sum_{m=2}^{n} \left\lfloor \left( \sum_{k=1}^{m-1} \lfloor (m/k) / \lceil m/k \rceil \rfloor \right)^{-1} \right\rfloor .$$

$$\frac{f(x+\Delta x) - f(x)}{\Delta x} \; . \; . \; f'(x) \qquad \text{as } \Delta x \; . \; . \; 0.$$

OCLC ROMAN 's scientific and mathematical fonts include math italic in text and bold weights, a font of common mathematical signs and symbols, including an upper case script alphabet, and a font which produces symbols of arbitrary size.

All of the OCLC ROMAN typefaces are available in sizes from 4 to 96 point and for use on devices with 100, 240, 300 or 480 dpi.

You can go from A to z.