**Inter-Office Memorandum**

| | | | |
|---|---|---|---|
| To | Whom it May Concern | Date | July 19, 1980 |
| From | Lyle Ramshaw | Location | Palo Alto |
| Subject | Cataclysm impact on TEX fonts | Organization | CSL |

XEROX

Filed on: [Maxc1]<Fonts>TexFonts.bravo
[Maxc1]<Fonts>TexFonts.press
[Ivy]<Fonts>Memos>TexFonts.bravo
[Ivy]<Fonts>Memos>TexFonts.press

Don Knuth just finished up the Metafont design of the family of fonts that will be used in the near future to typeset the second edition of Volume II of *The Art of Computer Programming.* I originally hoped to make those fonts available to PARC users of TEX as part of the font cataclysm announced on April 1, 1980. Unfortunately, I am not going to meet that deadline: the non-TEX portions of that font cataclysm happened on July 18, 1980, but the TEX portions won't happen for a little while yet. This documnt discusses the plans for the TEX font cataclysm that will occur later on. As part of this later update, we are planning to improve the manner in which font size is handled for Tex fonts.

**Physical Size versus Logical Size:**

What is the difference between a 9 point and a 12 point font in the same family? In the world of PARC font software, these two fonts are very closely related: in particular, you can get the 12 point font by merely scaling up the 9 point font by 1/3 in each dimension. Or you can get the 9 point by scaling the 12 point down by 25%. To put it another way, different sizes of the same family at PARC are strictly proportionally related. This proportionality has several advantages: first, the rasters for the characters can be produced by scaling their spline definitions, and then scan converting. In addition, a single table of width information can be scaled appropriately to give the widths of any size font in the family.

On the other hand, strictly proportional scaling is only an approximation to what a typographer really desires. It turns out that, as the height of a font decreases, the widths of characters should decrease less than proportionally. The thickness of the character strokes and the sizes of the serifs should also change non-proportionally as the height changes. The designers of the PARC software world realized this, of course, but chose to stick with strictly proportional scaling anyway, because of the programming benefits it provides. While proportional scaling is not perfect, it provides good results as long as the splines are not scaled up or down too far. When fonts of one family are needed that cover a wide range of sizes, the range can be broken into subranges, and a different set of splines produced for each subrange. In fact, this has already happened: TimesRomanD and HelveticaD are font families whose splines were drawn with the larger sizes in mind.

In Metafont, however, the width, stroke thickness, and serif sizes of each font are set independently to achieve the best typographic result. There is no assumption of proportionality. As an immediate consequence, the widths of each different size of each Tex font must be stored separately: this explains why Fonts.Widths files for Tex fonts are so large.

The non-proportionality of the Tex fonts wouldn't be as much of a hassle if it weren't combined

with another factor: people's desire to print magnified documents. Suppose that you are preparing a proceedings article that should appear in 9 point type. Also suppose that the printers of this proceedings will photographically reduces whatever masters you give them, say by 25%. If you are working with PARC fonts, you can allow for this reduction by choosing to print your document in 12 point type. Reducing 12 point TimesRoman by 25% will produce 9 point TimesRoman. But with Metafont, this is no longer the case. If you want the final result to be CMR 9, the stuff coming off the Dover should be CMR 9 scaling up to be 12 points in height, which is not the same as CMR 12.

The real truth is that fonts produced by Metafont have two different sizes: their *logical size*, and their *physical size*. The logical size is the size that Metafont should have in mind when it decides how to draw the character; the physical size is the actual height of the font as it comes off the printer. For the purposes of this memo, let's measure logical and physical size in *logical points* and *physical points* respectively. Note that printing with a font with sizes 10 logical points and 20 physical points, and then photographically reducing by 50%, is a way to make a Dover give you 768 pixels per logical inch instead of only 384. Of course, the price that you pay is that the Dover now prints on paper that is only 4.25 by 5.5 logical inches!

This conflict between logical size and physical size was already felt with the old Tex fonts, the ones currently on Clover. In addition to the regular CMR, there is a family called SLIDESCMR; the font SLIDESCMR 10 has sizes 10 logical points, and 14 physical points. Note that what has happened here, and in the TimesRoman and TimesRomanD example, is that one kind of size information is being encoded as part of the family name of the font. You have to do something a little bit exotic, since PARC font software only has one size field, and encoding in the family name is one easy thing. In the case of TimesRomanD, the size field gives the physical size, while the presence or absence of a D in the family name tells something about the logical size. In the case of SLIDESCMR, however, this convention is reversed: the size field tells you the logical size while the family name indicates the physical size.

Once the problem is described in these terms, it seems clear that it would be better to make a uniform convention about the use of the single size field provided by PARC software. We hereby propose that the size field should be used exclusively for the physical size of the associated font. Thus, TimesRoman and TimesRomanD are retroactively declared to be doing the right thing, while SLIDESCMR is retroactively declared a mistake. This convention has several advantages. For one thing, it makes it possible to produce a width table for CMR in 10 logical points that can be scaled in the standard PARC way to give the widths for any physical size. In addition, when the printing servers do font substitution, their matching algorithms assume that the size field really describes how big the font will be when coming off the printer.

We will take it as given, then, that the PARC size field for a font shall be used for the physical size. Note that this physical size need not be an integral number of points by the way. Suppose that you are setting math with Tex, using the standard logical sizes of 10, 7, and 5 points. If you want to magnify your output so that it can later be reduced by 25%, you want to use physical sizes 13.33, 9.33, and 6.67 points respectively. These fractional point sizes are likely to confuse humans, but, fortunately, they don't confuse most PARC programs: font sizes are actually measured by the software in micas instead of points, and micas are small enough that rounding to the nearest mica is irrelevant. Unfortunately, PressEdit is a source of difficulty here, since PressEdit does not implement the feature of Press file format which allows font sizes to be specified in micas. We will either fix PressEdit or implement a functional substitute before the TEX cataclysm occurs.

This example of fractional sizes above indicates that the user of Tex doesn't want to be concerned with the physical sizes of fonts. We propose that instead, the Tex user will specify the logical sizes and a magnification factor, where the magnification factor is a floating point number that expresses the ratio between physical and logical size. We plan to generate varying sets of TEX fonts in

varying magnifications: a full set at 1.0 (which is honest, afterall, and the universities would like); a fulll set at 1.1 (which makes CMR 10 the same physical size as TimesRoman 10), a partial set at 1.32 (which makes CMR 10 like TimesRoman 12), and a partial set at 1.54 (which makes CMR 10 like TimesRoman 14). The Tex user will specify the magnification factor by using the magnification parameter of TEX which is built-in as parameter 13, and accessible through \chpar13''. Thus, when a user specifies a font with the command \font a_CMR10'', the system will choose the family CMR in 10 logical points and (10*magnification) physical points. There will also by a \truefont'' macro that assumes a magnification of 1.0, just as the true distance units do.

Having chosen to put the physical size of the Tex fonts in the PARC size field, we are left with the job of encoding the family, face, and logical size of the Tex fonts in the remaining two positions: the PARC family name and the PARC face byte. What are these remaining PARC fields like? Well, the family name is a string that is mapped by an index at the beginning of each font dictionary or Press file into a corresponding eight bit code. The face byte is an eight bit field, of which PARC fonts currently use only the 18 different values from 0 through 21b, where these 18 possibilities correspond via a fixed mapping with the 18 different possible combinations of:

[Light, Medium, or Bold][Regular or Italic][Condensed, Regular, or Expanded].

(Update: A recent change suggested by CMU has added one more component to the face byte field: a three-valued component that distinguishes between different possible character code conventions. Thus, the 54 different face bytes in the range [0,53]=[0b,65b] are now spoken for.]

We can encode information in the PARC family name by just putting it into the string. Unfortunately, if we encode the Tex family, face, and logical size all in the family name, we will generate a huge number of family names; this could become a severe problem in the future, since no font dictionary can talk about more than 256 different families. This desire to limit the explosion of family names suggests that we should try and use the PARC face byte for something.

The first possibility that comes to mind is to use the PARC face byte to encode the Tex face. Unfortunately, Tex fonts come in all sorts of (by PARC standards) bizarre faces, including Slanted Sans Serif Quotation'', Unslanted Italic'', and the like. Thus, to encode the Tex face in the PARC face byte, it would be necessary to extend the fixed PARC face code. Given the arbitrariness of the Tex faces, the only way to extend the code would be to start a list somewhere, allocating face byte values to Tex faces as they arise. That list would become a critical sheet of paper, by the way, since any face code once assigned can never be changed, and can never be used for any other purpose. In fact, it seems undesirable to have to associate code values with arbitrary faces statically when the PARC family name mechanism allows codes values to be associated with strings dynamically. But putting the Tex face into the PARC face byte would be a feasible scheme, at least. If we did that, we would be left with the task of encoding the family and the logical size into the family name. Since there are likely to be fewer magnification factors than logical sizes, it would probably be better, actually, to encode the Tex family and magnification factor into the PARC family name: this would suggest family names such as ComputerModern-1.54'' and the like.

At present, however, I am leaning in favor of a different encoding scheme. The old Tex fonts had already chosen to encode the Tex face into the PARC family name, and that seems to work out rather well. Having handled the Tex family and face with the PARC family name, we are left with the chore of shoehorning the Tex logical size into the PARC face byte. I propose that face values in the range [54, 254]=[66b, 376b] be devoted to measuring the logical sizes of Tex fonts: the amount by which the face byte exceeds 54 will be the logical size of the Tex font measured in half-points. (This will allow logical sizes from zero through 100 points inclusive; the face byte value 255 will be left as an escape value.) For an example, the new version of the font Computer Modern Bold at 10 logical points and 14 physical points would be described by PARC software as having family CMB'', size 14 points, and face 74=54+2*10.

This perhaps exotic scheme has several advantages over the first scheme discussed: the Tex face,

which is basically a string, will be handled by the PARC mechanism best equipped to handle strings. The Tex logical size, which is a number, will be stored as a number instead of encoded into a string. Of course, allowing large values for face bytes will have some impact on PARC software, but a quick survey of knowedgeable people indicates that that impact will be smaller than you might think. The program PrePress already knows about these numeric face codes: face byte values in the appropriate range are input and output as floating point sizes in units of points. Both of the printing servers Spruce and Press, it turns out, don't do anything with face codes except compare them for equality; hence, no changes to either of those systems should be necessary. If you are a font wizard, and you see a reason why this encoding scheme won't work, please let me know. If I hear no bitches, it shall be done as I have said.

**Compatibility with the Past:**

The other major issue to be considered is compatibility with the past. The new Tex fonts are substantially different from the old ones. More than the fine detail of the character shapes has changed. The widths have changed a lot. The feel of the fonts has also changed, subtly in some cases, and blatantly in others: the upper case script alphabet in CMSY, for example, has been drastically redone.

Whenever a font revision occurs that involves changes in character widths, there is the possibility of old Press files not printing properly. Since the old Tex fonts have been around for more than a year now, there are probably a fair number of Press files in existence that depend upon those old fonts. As the standard PARC fonts got thinner and thinner over the years, one mechanism evolved for handling this problem. Those printers that run Spruce can save multiple tables of width information for a font in their font dictionaries. In addition, every Press file is now stamped with its date of creation. When Spruce receives a Press file, it checks this date; it then prints the file using the current rasters for the font, but spacing those rasters according to the widths that were in effect when that Press file was created. As a result, old Press files come out with different character shapes, but the lines still look justified.

Unfortunately, we can't use this multiple widths mechanism to handle the Tex font portion of the upcoming cataclysm because of the consequences of the logical and physical size decisions made above. Even if we could, the fonts have changed enough so that the results of printing an old Press file with the old widths and the new rasters might not satisfy you. But we can't do it anyway: existing Press files that are asking for CMR at 10 logical points and 10 physical points are asking for CMR with size 10 and face 0. According to the new conventions, you must ask instead for CMR with size 10 and face 74=54+2*10. Thus, the old Tex fonts and the new Tex fonts don't match, according to the tests of the printing servers. Therefore, we propose the following: we will simply leave the old fonts on the printer untouched for a while, to achieve some backward compatibility. After this period, the old fonts will go away, so that we can recapture the space that they require. I would guess that the old Tex fonts will stay on Clover for at least a few months after the cataclysm at least, but not for too much longer. I hope that most Tex users have been smart enough to save their Tex sources, so that dealing with old Press files won't be too big a problem. If you are the type of person who throws away Tex sources, consider yourself warned.

Even though the bulk of the old Tex fonts can stay in Clover's data base without any bad consequences other than making it bigger, the SLIDESCM* fonts are a different case. They are using up family names for what is, in the new scheme of things, no good purpose. I would like to throw those fonts away as part of the cataclysm, even though this would constitute a gross incompatibility with the past. My sense is that they have not been extensively used. And of course, the post-cataclysm Tex and Clover will have roughly equivalent fonts that fit into the scheme detailed above. If you object to the sudden disappearance of the SLIDESCM* fonts, please let me know.

One more minor issue to go! In general, it is impossible to use Tex fonts in Bravo. The standard Fonts.Widths file doesn't include the widths for Tex fonts, because they are too bulky to make that desirable. In addition, Tex fonts in general don't come equipped with Alto versions. Since the mathematical symbols in CMSY seemed too important to pass up, Leo Guibas took the 10 point version of the old CMSY and produced an Alto font for it, calling the result SYMBOL. Some undetermined number of Bravo users have now become dependent on the SYMBOL font. The script alphabet that is part of CMSY is one of the things that has changed the most in the new Tex fonts. My current plan is to leave the SYMBOL font alone; after all, someone might actually like those funny script characters!