UNICODE 対応の VF を作るために考えた方法（仮称：Multiple Fonts Decomposition）について忘れる前に書いておきます。汎用性があるような無いような。

2 バイトの文字を 16 進数で表すと 4 桁になります。最初の桁はフォント指定に用います。残りの 3 桁をそれぞれの和文フォントの文字に割り当てればよいのですが、TEX において扱えるのは 1 区 1 点の文字から 120 区 94 点の文字だけです。(それ故に、このような分解を考えなくてはならないのです。) また、3 桁の 16 進数のコードの集合を U とし、それぞれの和文フォントの持つ区点コードの集合を V_n，ここで $n = 0, \ldots, 15$ とします。U から V_n への上への写像 F が得られれば、フォントの分解が可能です。

ここで TEX の処理系による限界から、F は簡単なアルゴリズムである必要があります。(TEX では 16 進数の扱いに難がありますし、型変換なども用意されていません) いっぽう F^{-1} は OVP を作成するときに計算出来ればよいので（私は perl を使いました)、比較的難しくても大丈夫です。この F の構成方法がトリヴィアルでない点です。

写像 F の構成方法を説明します。3 桁の 16 進数は 12 桁の 2 進数で表現出来ます。これを上位 6bit と下位 6bit で分けます。6bit で表現出来る数値は 0–63 なので、それぞれを区点と思います。但し、そのままでは 0 区 0 点などのような区点コードになってしまいますので、それぞれに 16 を加えます。従って 16 区 16 点〜79 区 79 点までの区点コードに変換されることになります（各区の 80–94 点は存在しない)。ここで 1 ではなく、16 を加えたのは TEX がカテゴリコードをハードコーディングで指定しているためです。

逆に区点コードを 3 桁の 16 進数に戻すアルゴリズムは区、点をそれぞれ k, t とすると、$64(k-16)+(t-16)$ を 16 進数に変換した物が、元の文字コードの下 3 桁です。