Siamese TEX: Joining dvi Files at the Hip and Other Novel Applications of VF Files

Don Hosek
Quixote Digital Typography
349 Springfield #24
Claremont, CA 91711
714-621-1291
Internet: dhosek@ymir.claremont.edu

Abstract

When the utility of XPL files was revealed at the 1989 TUG meeting at Stanford University, and Donald Knuth announced that he would be working on an updated format for use with TEX, it was expected that this new format, VF, would become quickly and widely accepted in the TEX community. As it turns out, nearly two years after the creation of the format, the use of VF files is still fairly rare. This is due partly to lack of understanding of what can be done with VF files and partly to a lack of tools for implementing these capabilities. This paper will seek to fill both gaps: by presenting an introduction to what can be done with virtual fonts and also by describing some recently created utilities to facilitate the implementation of their potential.

What are VF Files?

First off, let's open up the acronym and point out that VF stands for "Virtual Fonts." There are some who would claim that this term is a little misleading in the context of other computer science technology and prefer the term "Composite Fonts." As a non-computer scientist, I prefer to stick with the term "Virtual Fonts" myself, mostly because it matches the acronym better.

Now that we have that formality out of the way, perhaps it is time to ask what it means for a font to be "Virtual" or "Composite." It means that what TEX thinks is a single character is not necessarily that same single character to the printer. Some dvi drivers have had a limited version of this capability built into them by necessity: for example, many dvi-to-HP LaserJet converters will map character codes to different positions (to allow for restrictions on permissible character codes on fonts) and will send larger characters as bitmapped graphics or "tiled" pieces of character. However, this capability is rarely within the control of the user.[1]

With the VF format, we have an opportunity to handle many useful features in a way that *could* be device-independent.[2] In fact, with VF files, we find that not only can we handle remapping of fonts rather easily, but we can also build composite characters: by combining characters from the same or different fonts, and also by mixing any elements which may appear in a dvi file such as rules or \special commands as well. There are also device-

[1] Tom Reid's TEXROX drivers came close to implementing some version of VF support in its ROXDEX files which at least allowed manual control over the mapping from TEX character code/font combinations to Xerox character code/font combinations. However, this facility was not terribly robust and an early experiment in creating a Times Caps/Small Caps font revealed that the TEXROX was expecting better behaved ROXDEX files than I was creating. In particular, a given Xerox font could only be referenced from a single TEX font in any dvi file causing problems when the 10pt Times Roman was accessed in both the Roman and Caps/Small Caps fonts. Tom Reid later adapted the program to support this sort of tinkering, but by then I had left my beloved Xerox 8790.

[2] The current state of matters requires the "could" in the preceding sentence; Appendix A has details on precisely what is supported by the current software.

dependent applications for VF files which we will discuss in more detail later.

TEX 3.0 and VF Files

Since TEX generally doesn't know much more about a font than the amount of space that each character takes up, it isn't even aware that VF files are being used in a TEX run. Support of VF is left entirely up to the dvi-to-output converter.

Incidentally, I think that this is a big design flaw. Had VF support been built into TEX itself, TEX would have become much more versatile. For example, the problem of hyphenating accented characters could have been eliminated once and for all since one would have been able to build accented characters "on the fly." In addition, the ability to have arbitrary remapping of fonts when TEX is run would have solved the notorious "code page" problem. However, since Knuth was in a hurry to finish TEX 3.0, we'll forgive him this oversight.

Code Pages and Remapping

We'll ignore the problem of mapping the input character code to the output code (perhaps I'll write a brief *TUGboat* article on my experiences on the topic). Instead, we'll focus on a rather specific problem: a physical font may not have the mapping of character codes in it which we would want to modify to conform to a scheme of our own. For example, if we use Personal TEX's PTI Fontware Interface to generate .pk files from Bitstream fonts, we get three 128-character fonts which are not quite in an arrangement suitable for our desires. Most likely, we would want a 256-character font corresponding to the proposed standard developed at the 1990 TEX Users Group meeting at Cork [1].[3] To effect this we can employ VF files to handle the remapping of character codes, as seen by TEX, to the character codes that are actually used in the font.

One way to accomplish this would be to create a VPL file by hand (a VPL file is a VF file converted to a mnemonic form readable by humans). As it turns out, this isn't too bad; Figure 2 shows how a remapping of this sort might look. However, in practice, this can get rather tedious since we need to give metric information for every character.[4] Even without this hindrance, it would still be overly tedious for a font where the remapping involves fairly direct remappings, e.g., for a small caps font, where almost the entire font would be mapped directly except for the lowercase letters which would be mapped from the uppercase in a smaller font.[5]

```
(MAPFONT D 0
   (FONTNAME beckman)
   (FONTCHECKSUM 0 10537600616)
)
(CHARACTER 0 15 (comment quotesingle)
   (CHARWD R 366)
   (CHARHT R 816)
   (MAP
      (SETCHAR 0 47)
   )
)
```

Figure 2: An extract from a VPL file showing how remapping of characters could be accomplished. The above sample was generated by Tom Rokicki's AFM2TFM.

The REMAP **utility.** To simplify this task, I wrote REMAP, which provides a far simpler format for specifying this most common application for virtual fonts. A REMAP input file begins with a series of lines indicating the fonts that are used in the format:

FONT *font_number font_name* [*optional_scaling_factor*]

The *font_number* is any number between 0 and 15.[6] The most commonly used font should be assigned number 0 as in a VPL file; we will see that this helps cut down the coding. The *font_name* is the font as it is known to TEX, e.g., cmr10; this could be the name

[3] Or not. Speaking from the font designer's standpoint, I find that the Cork standard has an inadequate number of vacant font spaces for face-specific characteristics. For example, five "f-ligatures" are provided, but not all are necessary in all fonts, but no space is reserved for an f-j ligature which is useful for typesetting Scandinavian languages. For some classical designs, other characters are necessary as well, e.g., ligatures for c-t and s-t, long s, and others.

[4] A fact not adequately documented, incidentally.

[5] This is actually a rather simplistic view. In a high-quality small caps font the weights of the small caps are adjusted to correspond to the weights of the lowercase letters of that size. With most modern digital type technologies, where a single outline is linearly scaled for all sizes, using 8pt caps for the lowercase in a 10pt caps small caps font ends up with small caps that are too light for the surrounding text.

[6] The upper limit of 15 is arbitrary and was intended to keep memory requirements low

```
FONT 0 BS0011
FONT 1 BS0011 800
RANGE 0 127
DATA DECIMAL
0:0 1-96 1:65-90 123-127
END
```

Figure 3: Sample REMAP input file. The specification `0:0` at the beginning is intended to show the format of a single character mapping. In practice, the data would begin `0-96`.

of another virtual font, although such nesting can be dangerous. Finally the *optional_scaling_factor* is an integer which gives the scaling factor in the same terms as the `scaled` keyword in TEX: e.g., 1000 refers to no scaling, 500 gives a 50% scaled face, and so on.

After all the `FONT` statements have been declared, a line must appear which gives the range of character codes which define the extent of the font. This serves as a simple checksum against typographical errors in the last segment of the REMAP file. Its format is:

`RANGE` *first last*

Finally, the remap data is provided. All numbers must be in the same radix but a choice of hexadecimal, octal or decimal is provided. The remap data consists of a font number – character code pair for each character to be remapped. If the font number is 0, it may be omitted. Each pair is joined with a colon, and pairs can be separated by one or more spaces or a new line. A contiguous range can be specified by listing the first and last character code separated by a hyphen. An unused character position is indicated with `XX`. The data begins with `DATA` followed by one of `HEX`, `OCTAL` or `DECIMAL` (the default, if none of the choices is listed, is `HEX`). At the end of the data and the file, `END` should appear on a line by itself. Figure 3 shows a sample of how this might appear for a small caps font.

One feature of REMAP which is not readily apparent from the above discussion is that kerning and ligatures from the fonts being remapped are preserved as far as possible. For example, a kern between the A and V of font 0 would be preserved as would the corresponding kern between the A and V of font 1. However, no kern would automatically be inserted between, say, the A of font 0 and the V of font 1. Also, kernings and ligatures for characters not included in the remapped font would be ignored, so the user need not worry about getting "DIFFICULT" instead of "DIFFICULT." If the user feels the need to add or delete ligatures or kerns explicitly, this can be accomplished through the keywords `LIG`, `NOLIG`, `KERN` and `NOKERN`.

VF Files, POSTSCRIPT Fonts, and My Previewer

As was mentioned earlier, `VF` files can be a useful tool for dealing with *any* device-dependent typefaces. Tom Rokicki's AFM2TFM converter, for example, uses `VF` files for remapping character codes and creating small caps versions of the fonts. However, this technique still relies on there being a font on the printer somehow associated with a set of `tfm` dimensions on your computer.

If we want to preview a `dvi` file which refers to these fonts, we have two options: the first would be to have `.pk` files which match the POSTSCRIPT fonts. This is certainly a possibility and there exist from various sources many options for creating these files.[7] Any necessary remapping of character codes can be handled using the REMAP features described above. This, however, is not interesting.

A more interesting approach would be the case where an exact preview is not necessary and it would be adequate to use, say, Computer Modern to give an approximate the appearance of the printed document in screen preview or possibly even using POSTSCRIPT fonts to create proofs of a document which is to be ultimately printed using native fonts on a Linotronic typesetter. Mapping the characters to appropriate places, as noted above, is trivial; however, problems will be encountered in proper spacing of the letters if the metrics for the font do not match. `VF` fonts can be used to remedy this situation by adding or subtracting appropriate amounts of space from the sidebearings to get the character widths to match. This is done with the `SHADOW` command in REMAP which causes all characters in the virtual font to have the same metrics as the characters with the corresponding codes in the font mentioned in the `SHADOW` command. This feature can also be used to create "invisible" fonts like those used by SLITEX. Invisible characters can be created by referring to characters in an unspecified font number: an example of this appears in Figure 4. This approach gives a slight storage advantage over the `tfm/.pk` strategy,

[7] Perhaps the best option in the MS-DOS world is to use Adobe Type Manager to create `CHR` files which can be converted into TEX-style fonts with the CHtoPX and PXtoPK utilities supplied with the public domain emTEX. This will allow any POSTSCRIPT font to be printed or previewed just like a METAFONT-generated font.

```
SHADOW ptmr
RANGE 0 255
DATA DECIMAL
0-255
```

Figure 4: A REMAP input file to create an "invisible" version of PostScript Times.

which is standard for the SliTeX fonts, and there already exist VF fonts distributed with some versions of Tom Rokicki's DVIPS for this sort of use.[8]

Device-dependent virtual font files. At this point, it would be worthwhile to point out that VF files fall into two categories. A simple remapping of the characters in a font such as that described in the introduction to REMAP would fall into the category of non-device-dependent VF files since they would be used with any output device. On the other hand, VF files created using the SHADOW feature of REMAP for proofing purposes would be device-dependent.

The classification into the two categories is not always self-evident. For example, the VF files created by Tom Rokicki's AFM2TFM would *not* be device-dependent! As it turns out, the remapping of characters performed by these VF files is still needed for VF files which will be used in the proofing stages.

Accented Characters

Another useful application of VF technology is the ability to create pre-accented letters. This is the only way to have TeX automatically accent words containing accents like "Explosionsgefährlich".[9] REMAP provides this capability with the ACCENT command which allows one to define accenting capabilities. The algorithim used for constructing accented characters is identical to that used for \accent by TeX. At the time of this writing, facilities for diacritics below letters (e.g., ç) are under development.

"Joined at the Hip" Explained

One other program was written as a sort of prolog to the work done to REMAP as described above. SIAMDVI is a program which takes a dvi file and creates a VF file in which each page in the dvi file is represented by a single character in the VF file. By default, the character dimensions are determined by the locations of print on the output from the dvi file, but this can be altered through the use of \special commands. The program had originally been conceived as a clumsy way of handling some of the features of REMAP, but use and discussion of its potential have revealed the following possible applications:

- Simple dvi inclusion. Actually, this approach is somewhat more sophisticated than that provided by Michael Spivak's DVIPASTE [3] or Stephan v. Bechtolsheim's DVI2DVI [4] since it is possible to load the virtual font scaled by some factor to include reduced views of output pages.
- The previous item can be expanded on with some simple macros to allow TeX to handle composing signatures. The current version of SIAMDVI limits this to books of 256 or fewer pages (because of the limitation on the number of characters which can appear in a single TeX font) but a future release will allow longer documents to be remapped. This has the advantage over many processors for signatures in that the positioning of pages can be adjusted slightly to compensate for the thickness of the sheets of paper in the final printing.

  If the output driver supports 180° rotation of characters, full signature pages could be composed in this manner.
- Using Alan Hoenig's METAFONT and TeX code described elsewhere in this proceedings issue, university seals can be typeset as single characters.

These are just a sampling of the possible uses of SIAMDVI. I had originally viewed it as more a novelty than a genuinely useful product—that evaluation has changed.

A Device Drivers Supporting VF Files

At the time of this writing, the following drivers were the only ones which I was aware of which supported VF features:[10]

[8] The files I have were in the MS-DOS distribution and are of unknown origin.

[9] This is because of a design decision in the TeX program. Knuth has justified TeX's deficiency in this regard as an encouragement for font designers to provide pre-accented characters. METAFONT-based fonts designed on this basis are just becoming available.

[10] Please be aware that any subjective comments in the list below are exclusively my opinion and are based on direct experience except where noted. I apologize for any inaccuracies.

- ArborText drivers updated since 1990. Some drivers in the ArborText collection undergo infrequent revision (e.g., `dvixer`) and so may not yet have this feature. However, since the `VF` format is based on Arbortext's `XPL` format for the DVIAPS driver, they have had a head start on implementing the features in their drivers. I have not used versions of these drivers containing `VF` support.
- DVIPS by Tom Rokicki. This public domain `dvi`-to-PostScript converter is the first public domain driver supporting `VF` to include source code. Also included with DVIPS is a program, AFM2TFM, written by Donald Knuth [2] and modified by Tom Rokicki, which uses `VF` files in creating remapped versions of PostScript fonts with support for ligatures and other convenient features. DVIPS runs on Unix, VMS and MS-DOS.
- The emTEX drivers. These drivers, usually bundled with the public domain emTEX for MS-DOS, provide excellent functionality for the user, although I have never bothered with the font library support which seems cumbersome to me.
- Radical Eye drivers for AmigaTEX. All programs have support for PostScript programs (even on non-PostScript devices!) and so come with an AFM2TFM program based on that with DVIPS (see above).

There are also two programs available for converting a `dvi` file which contains references to `VF` files into a `dvi` file with the references expanded into "clean" code which could be translated by any `dvi` processor. These are:

- DVIcopy by Peter Breitenlohner. MS-DOS, Unix, VM/CMS.
- DVIvfDVI by Wayne Sullivan. MS-DOS only.

B Status of the Programs

At present, the programs exist only in ugly VAX C code. Before release, the code will be translated into `CWEB` with change files for Turbo C and VMS available on release.

References

[1] Ferguson, Michael J. "Report on Multilingual Activities." *TUGboat*, 11(4), pages 514–516, 1990. [Page 516 of the report is a full-page table depicting the character set.]

[2] Knuth, Donald E. "Virtual Fonts: More Fun for Grand Wizards." *TUGboat*, 10(1), pages 13–23, 1990.

[3] Spivak, Michael, Michael Ballantyne, and Yoke Lee. "Hi-TEX Cutting & Pasting." *TUGboat*, 10(2), pages 164–166, 1989.

[4] Bechtolsheim, Stephan v. "A `.dvi` File Processing Program." *TUGboat*, 10(3), pages 329–332, 1989.

[5] Youngen, Ralph, William B. Woolf, and Dan C. Latterner. "Migration from Computer Modern Fonts to Times Fonts." *TUGboat*, 10(4), pages 513–519, 1989.