

integers for the hardware? (I recommend truncation since it is fast, and the user may change it to rounding by adding .5 to the translation components in the CTM.) Second, what is the precise orientation of bitmap characters generated by `imagemask`, with respect to the *current point*. I suggest that the current point should coincide with the extreme lower left-hand corner of the rendered image. That is, when the CTM is as described in the earlier example, the current point should identify the lower left-hand corner of a pixel, and this pixel should be overlaid with the lower leftmost pixel in the bitmap image. (A related issue is that when the coordinate system is inverted vertically, the current point should coincide with the extreme upper left-hand corner of the image — this does not appear to happen with our printer.)

A description of how the PostScript software is structured, distributed, and implemented on specific devices would also help applications developers to understand its operation. My guess is that the basic PostScript interpreter is provided by Adobe Systems, and each device manufacturer writes their own driver, but this is only a guess. Perhaps device manufacturers tell Adobe how their machine works, and later receive a fully-configured interpreter in machine-code form. Just how much does the device manufacturer do, and by implication, how much can be expected from a given PostScript-compatible product?

The issues raised in this article came up in the course of research into musical score setting using  $\TeX$ . I have been working with a modified DVI-to-PostScript driver which allows inclusion of arbitrary PostScript code into  $\TeX$  source material using the `\special` primitive. The idea is to use the power of PostScript to draw all the variable elements of musical material (e.g. note stems of variable length but fixed width). The lack of a definitive explanation of how PostScript's graphic primitives work at the device level forced me to spend a great deal of time writing tiny PostScript programs and examining the results — with a microscope! — to figure out what the printer was doing. I needed the microscope only to measure the *extent* of various inaccuracies in size and position — at 300 dpi the *existence* of these inaccuracies is immediately obvious to the naked eye.

## References:

- [1] Lamport, L.  *$\TeX$  Output for the Future*. TUGboat **8,1** (April 1987).
- [2] Adobe Systems Inc. *PostScript Language Reference Manual*. Addison-Wesley, Reading, Massachusetts. 1985.

## Index to Sample Output from Various Devices

Camera copy for the following items in this issue of TUGboat was prepared on the devices indicated, and can be taken as representative of the output produced by those devices. The bulk of this issue has been prepared at the American Mathematical Society, on a VAX 8600 (VMS) and output on an APS- $\mu$ 5 using resident CM fonts and additional downloadable fonts for special purposes.

- Apple LaserWriter (300dpi): ArborText advertisement, p. 110.
- $\TeX$ nology, Inc., advertisement, p. 103.
- Canon CX (300 dpi): Georgia Tobin, *The ABC's of special effects*, p. 15.
- Compugraphic 8600 (1301.5 dpi):  $\TeX$ t1 advertisement, p. 106.
- HP LaserJet (300dpi): Personal  $\TeX$  advertisement, p. 99.
- Linotronic 100 (1270 dpi): Design Science advertisement, p. 105.
- Kellerman and Smith advertisement, cover 3.
- Micro Publishing advertisement, p. 101.
- Xerox 4500 (300 dpi): Greek sample text, in Silvio Levy, *Using Greek fonts with  $\TeX$* , p. 22, as indicated.