PostScript Quick Tips: Permanently Downloaded PostScript

I recently implemented software for an Adobe Exhibit at The Tech Museum of Innovation in San Jose. This exhibit uses a video camera to take a picture of the visitor. The Photoshop Magic Wand tool is used to replace the background with a background chosen by the visitor. Daystar Digital's PhotoMatic provides scripting capability so that a customized user interface can be used to control Photoshop.

One of the requirements was to print a brief description of the exhibit on the same page as the visitor's photograph. Trying to import the photo from Photoshop into a page layout program would have complicated the scripting effort. The solution was to permanently download the required PostScript code into the printer (a Tektronix Phaser 540 color laser printer). This made it possible to print the entire page directly from Adobe Photoshop.

Permanently downloading PostScript code into the printer is only feasible when a dedicated printer is available. The downloaded PostScript code would most likely interfere with other users or other applications sharing the printer. At a minimum, the downloaded PostScript code consumes some of the memory in the printer that would otherwise be available for downloaded fonts or other uses. Note that the downloaded PostScript code remains in the printer until the power is cycled or the printer is reset. If the PostScript code is required on an ongoing basis, the code must be downloaded each time the printer is turned on. (If the printer has a hard disk, it is generally possible to download the PostScript code to the hard disk, so that it remains in the printer after the power is cycled. In most cases, however, it is sufficient to download PostScript code into the printer's RAM.)

```
serverdict begin 0 exitserver
% define showlogo and showtext here
/image {0 -0.4 translate
   systemdict /image get exec} bind def
/showpage {initgraphics showlogo showtext
   systemdict /showpage get exec} bind def
```

For the Adobe demo, the downloaded PostScript code performs the following functions:

- Redefines the PostScript image operator to move the image from the center of the page to the top of the page. (Normally, Photoshop centers the output on the page.) If the image had not already been the correct size, the downloaded PostScript code could have scaled it as well.
- Defines PostScript procedures to display the Adobe logo and the descriptive text.
- Redefines the PostScript showpage operator to display the Adobe logo and the descriptive text before printing the page.

Figure 1 shows a portion of the Downloaded PostScript code. The first line specifies that the downloaded code is to be remain in the printer after the download is complete. The second line is a comment which needs to be replaced with the actual showlogo and showtext procedures. The next two lines redefine the image operator, and the final two lines redefine the showpage operator.

The showlogo procedure uses a modified EPS (Encapsulated PostScript) logo. EPS files normally generate marks on the current page. What is required, however, is a mechanism to generate marks on pages that will be printed in the future, not at the time that the PostScript code is downloaded to the printer. This requires that the EPS artwork be placed into a procedure which can be executed at a later time. There are two potential problems.

First, only the simplest EPS artwork can be defined in a single procedure. Most artwork is too complex (contains too many tokens), and will overflow the operand stack if an attempt is made to enclose it in a procedure. In general, it is necessary to decompose the EPS artwork into multiple procedures.

Second, many EPS files contain procedures or resources in their prologue; these may conflict with the prologues from other logos, or from the LaserWriter driver. (This is analogous to attempting to simultaneously download incompatible versions of Laser Prep.) Therefore, it is often necessary to place each EPS graphic into a separate dictionary, so that the procedures and resources do not conflict with each other.

Because of these problems, the task of converting EPS graphics to procedures is a task best left to an experienced PostScript programmer.