The Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Working Paper 299 July 1987

# Merging Illustrations and Printing on Big Paper

Gerald Roylance

**Abstract**

How to guide for some of the printing utilities in the AI Lab. Describes how TEX files are processed and how some illustrations may be merged into the final copy. Also describes how to use TEX to print on 8.5x14 (legal) and 11x17 size paper.

# 1 Life in the Fast Lane

We know that you love the smell of rubber cement and other industrial adhesives, but we want to free you from the hours of tedium caused by cutting and pasting illustrations into your papers and your columns onto model paper. We will tell you how to print your TEX output on big sheets of paper and merge in your favorite illustrations. No part of this document required scissors or glue.

The most important thing you should know is that even though it is possible to merge illustrations and print on large paper, it is not something that is guaranteed to work. There are many places to spin out along the way. If what you're trying to do doesn't work out, then **you are on your own**. The information in this document is believed to be accurate, but it is sketchy and subject to change without notice. Perhaps the faint hearted shouldn't venture forth. But take heart, because some things are simple and won't take much effort at all.

One of the first trouble spots we run into is keeping the variety of TEX dialects straight. In this paper, TEX will usually refer to all of the dialects, but at times we have to distinguish between plain TEX[5], LATEX[6], and Brotsky's YTEX[2]. The text tries to be clear about these distinctions when it is important. The different dialects also have their peculiar problems. If you use YTEX with something other than a 10 point type size, then you should expect to have problems. YTEX uses the magnification parameter to attain these other point sizes, and consequently all your dimensions get magnified. When you ask for a 1 inch length in YTEX, you probably won't get something 1 inch long.

For those of you in a hurry, it is short circuit evaluation time. This document not only contains some quick advice on how to make some easy things work, but it also contains some technical information to help you tackle some more difficult problems. Many of you aren't interested in the difficult problems (and shouldn't be until you get burned by the simple approach), so you will want to go right to the relevant section for what you want to do.

- If you are in a hurry and only want to merge pictures made by Illustrate into your document, then see section 2.1.2.
- If you are in a hurry and only want to print on larger paper, then see section 3.

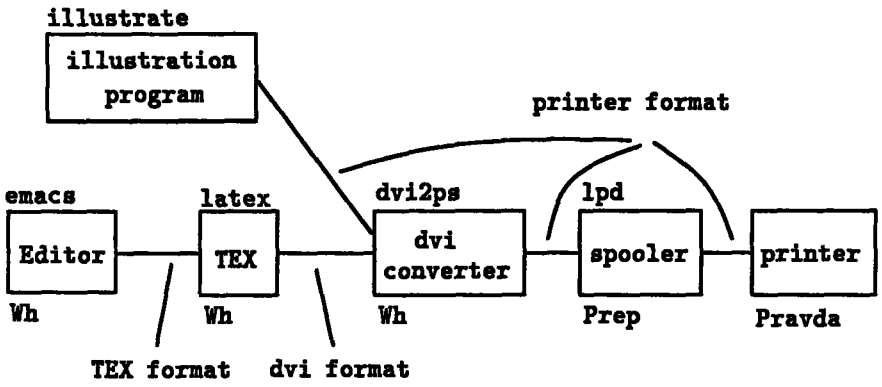


Figure 1: Printing Process

- If you are in a hurry and want to merge pictures and want to print on larger paper, then either slow down or use your scissors. It is easy to do one or the other, but both at the same time requires some care.
- If the printers cannot print your whole dvi file, then you have to subset the dvi file. See section 4.2.

Bugs with the software should be reported to Bug-Printer@OZ. If a printer is broken, you should call it in. The phone numbers are next to the printers.

Advice is 60 cents. Exact change. No pennies.

## 1.1 TEX Process Description

The process of printing a TEX file is an involved one. It gets more involved when pictures are merged or the paper size is unusual. The rest of this document will talk about the components of this process: TEX special commands, TEX, dvi converters, spoolers, and printing engines. This section is how they all fit together.

Figure 1 shows a simplified process diagram of printing a TEX file. Boxes represent processing units—programs or machines. The arcs represent data links; sometimes the links are files and sometimes the links are wires. The diagram also includes where different files are merged.

The first step is using an editor to prepare the TEX source file. Special commands, which are instructions to the dvi converter, are entered into the TEX source. The output of the edit step is a file with the extension (or type) of `tex`.

The second step is running TEX on the source file. TEX formats the input source and outputs the result as a dvi file. The file, which has the extension `dvi`, is a device independent representation of characters and their positions. TEX also passes through, after some minor processing, the special commands.

The third step is converting the dvi file to a format a printer understands. While a dvi file is a device independent format, no printer actually understands it. The printers in the AI Lab understand two printer formats: PostScript[4,3] and QUIC[1]. It is during this conversion step that the special commands are interpreted. If the special command calls for inserting a file, the file is inserted now. Notice that TEX and the dvi converter do not necessarily run on the same machine. Consequently, the files that were accessible to one machine may not be accessible to the other.

Once the printer format file (a PostScript or a QUIC file) has been generated, it is sent to the printer spooler. The spooler queues the printing request and sends it to the printer when the printer is available. The spooler does not look at the file contents[1]; it merely sends it to the printer.

Finally, the printer reads the device dependent printer format, converts it, and paints the pixels on sheets of paper. The printer also selects the appropriate feed trays and/or paper sizes; you, however, have to load the right paper into the printer before printing starts.

## 1.2 The Cookbook Method

There are many possible combinations of doing things and figuring out the easiest way to do them is quite a chore. It's also something that you can only do intelligently after you've tried out some of the alternatives. So here are some recommendations about making your document that will make life easiest. Of course, the situation will change and in the future there will be better ways.

- Use PostScript illustration formats.

[1] There can be some confusion here. The program that sends files to the spooler, `lpr`, can be told to route the file through the dvi converter. The spooler is the program `lpd`.

- Use a Unix machine to store your files. You will need to run some Unix programs to print your file, so keeping everything on a Unix machine to begin with avoids problems with files.
- Put all the source files for your document in one directory. That includes all of your illustration files, too. Having all the files in one directory simplifies the problems with filenames. Instead of giving fully qualified file names that include the host, device, and directory, you need only supply the file name and file type (eg `foo.ps`).
- Use the macros at the end of this document. They can be `\input` into your TEX source file.
- Connect (`cd`) to the directory and run TEX.
- Set up the printer (section 5).
- Connect (`cd`) to the directory and run `dvi2ps` piping the result through `lpr`. See section 4.

## 1.3 Current problems.

Although a somewhat consistent view of the world is held by dvi converters and previewers, there are still many problems. Some things just don't work, sometimes you have to trick them to work, and sometimes we **suspect** that something won't work but we haven't checked it out yet. So right up front we are going to tell you the bad news as best we know it.

The dvi converters and printers are not very robust. The dvi converters assume that the printer can digest an arbitrarily large file, but the printers frequently gag. The QUIC printers can print the largest files, the PS2400s can print the next largest, and the PS800s and LaserWriters print the smallest. The dvi converters that run on the Lisp Machines take a lot more time and produce much larger files for the printers to digest. If your document is more than 20 pages, there is a good chance the printer cannot swallow the Lisp Machine's output; use the Unix converters instead. If you still lose, you'll have to dvi convert the document in parts (see section 4.2). Symbolics needs better hardcopy code. The dvi converters should clear out the printers memory every 50 pages or so, but they don't. Writing code to fix this problem would require 4 days of work.

There are some horrible interactions that concern the TEX magnification parameter. Dimensions may not turn out the way you want them. While magnification does work on the TEX document itself, the printers may not have the magnified fonts available. You may have to adjust the dimensions of your illustration and your margin parameters to account for the magnification. Though it makes sense to magnify both illustrations and text according to the TEX magnification parameter, the current special commands will not do this transparently. Though we haven't checked, the output of Illustrate may not be scalable. To avoid trouble, you should not magnify your document. Two days of coding effort here might introduce more reasonable special commands that would respond correctly to magnification.

The paper size special commands have different semantics on different printers and maybe even different dvi converters. Paper size commands are sticky on QUIC printers, but are not sticky on PostScript printers[2]. You will have to run the PostScript dvi converter by hand to get around this problem. These are serious faults and need to be fixed soon; it is probably 2 days worth of effort.

There must be a better way of inserting the paper size special commands into LATEX.

We should have a library of TEX macros and LATEX document styles that people can use. Specifically, we should have macros for theses, memos, and publications such as AAAI and IJCAI. Maybe we already have some.

# 2 Merging Illustrations.

We can merge 4 types of illustration[3] formats: QUIC, PostScript, image, and mugshot. This discussion is centered around merging PostScript because it is the most common case; if you are going to use a different format, you should still read about merging PostScript. If you are just interested in merging Illustrate files, then you only need section 2.1.2.

[2]This may only be true of the Unix dvi converter; the Lisp Machine dvi converter hasn't been tried.

[3]Illustration here does not necessarily mean the output of the program Illustrate.

```
\special{psfile=<filename>
         hsize=<dimension>
         vsize=<dimension>
         hoffset=<dimension>
         voffset=<dimension>
         hscale=<number>
         vscale=<number> }
```

Figure 2: `psfile` special command

## 2.1 Merging PostScript

Of the two major printer formats the AI lab uses (PostScript and QUIC), it makes the most sense to merge PostScript format illustration into your text. Most QUIC format commands use absolute coordinates and so must be redrawn every time the figure moves. Absolute coordinates also mean that QUIC illustrations cannot be magnified.

The first requirement is that we have a PostScript file to merge. This file could be generated by a program such as Illustrate or generated by hand. The PostScript file is actually merged when the dvi file is converted to the format the printer takes and not when you run TEX. Consequently, the PostScript file must be accessible to the machine that runs the dvi converter and not the machine that runs TEX. See more about the file access problem below.

The dvi converters on Unix and on the Lisp Machines understand the `psfile` special command[4]. For more information about the dvi converters, see section 4. In addition to taking a filename, the `psfile` special command also takes several positioning and scaling options (figure 2).

Here's what the `psfile` command does. Let position of the special command on the printed page be an absolute (x, y) (this position thus includes any scaling done by the TEX magnification parameter). That absolute position is adjusted by an absolute (`hoffset`, `voffset`) (these numbers are not scaled by the TEX magnification parameter or by `hscale` or `vscale`). That position on the page is taken as the origin of the inserted PostScript file. All of the dimensions in the PostScript file are then scaled by `hscale` and

[4]For a discussion of special commands, see the TEXBook[5].

```
(load "OZ:<GLR.PSTRM.PLOT>PLOT.BIN")
(load "OZ:<GLR.PSTRM.PSCRPT>PSPOST.BIN")
(defun papers-sine ()
  (with-open-pstrm
      (stream (pspost-open "prep:/u/glr/sine.ps"
                           0.0 4.0 0.0 3.0))
    (plot-fcn 200 #'sin 0.0 6.3 stream)))
(papers-sine)
```

Figure 3: Lisp code to plot the sine function.

`vscale`. For more information do `man dvi2ps` on a Unix machine.

We want TEX to treat our illustration exactly like a (big) character so that TEX will make room for it on the page and the spacing around the illustration will be consistent with other spaces in the document. Special commands, however, are boxes with zero width, height, and depth. All of the commands given in this paper conspire to make the appropriate box dimensions. Furthermore, the depth (the amount the box extends below the baseline) of all of the boxes is set to zero.

But that only takes care of one of the coordinate systems. We have to position the `psfile` special command somewhere relative to this box in such a way to make the picture end up inside the box. And it is here where most of the trouble starts. TEX uses a left handed coordinate system with the origin at 1 inch down and 1 inch over from the top left corner of the page. Some pictures will use the lower left corner as the origin of a right handed coordinate system. Others will use the upper left corner as the origin of a left handed system.

The simplest convention is to put the special command in the lower left corner of the TEX box. For convenience, we define the command `psbox` to insert a PostScript file and to build a (hopefully) surrounding box. See section 7 for the definition of this macro.

As an example, let's insert a plot of the sine function. The lisp code (along with a plotting package) shown in figure 3 writes a PostScript file that is a graph of the sine function. The size of the plot is 4 inches wide by 3 inches tall and its origin is the lower left corner of the page. That graph is

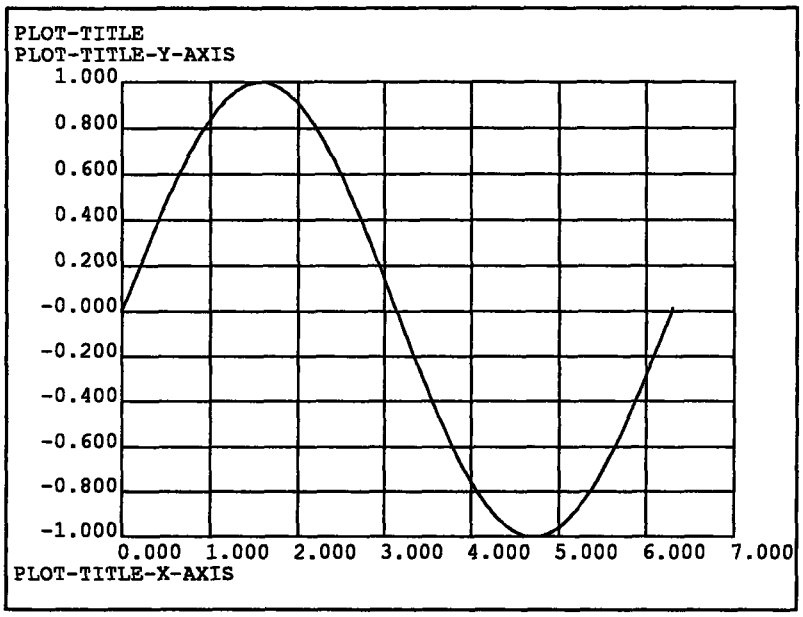


Figure 4: Plot of the sine function.

inserted into this file with the LaTeX commands:

```
\begin{center}
\fbox{\psbox{4.0in}{3.0in}{/u/glr/sine.ps}}
\end{center}
```

The finished result is shown in figure 4. The LaTeX `fbox` command outlined the box where TeX expected the graph to go; it's a useful debugging command when your illustration isn't going in the right place. A plain TeX equivalent of `fbox` is given in figure 5.

In general, for different conventions for aligning the illustration with TeX's idea of its position, we will have to adjust the offsets. These offset adjustments should not be made on a per figure basis; the insertion macro should take care of everything as long as the width and height of the figure are given correctly. For more information about alignment, see section 2.1.2.

```
\def\fbox#1{%
  \vtop{\vbox{\hrule%
              \hbox{\vrule\kern3pt%
                    \vtop{\vbox{\kern3pt#1}\kern3pt}%
                    \kern3pt\vrule}}%
        \hrule}}
```

Figure 5: Plain TEX Version of `fbox`.

### 2.1.1 Automatic Sizing

Many PostScript files follow the file structuring conventions laid down by Adobe[3]. Under this convention, bounding box information is written at the head of the file. This information can be read by TEX and used to align and size a picture. The macros do not currently exist for LATEX.

There are two versions of the macro; one, `psadobe`, will magnify the illustration by TEX's magnification parameter. The other, `psadobenomag`, produces the illustration at actual size. Each macro takes one argument, the name of the file. The file must be accessible to both the machine that runs TEX (so it can read the file header) and the machine that run the dvi converter (so it can insert the file).

```
\psadobe{filename.ps}
\psadobenomag{filename.ps}
```

As I was finishing this report, we found out about a polished system by Trevor Darrell that does automatic sizing. It is called `psfigtex` and works for both TEX and LATEX, but it requires a different dvi converter than we currently have (the system uses a `psfig` special command). Darrell alludes to a `bbfig` utility that calculates the bounding box of a PostScript file that doesn't have one. A copy of this system is in

`prep:/u/glr/psfigtex/`

### 2.1.2 Illustrate

The program Illustrate can be used to make illustrations on the Lisp Machine. The problem is getting your illustration to end up on the right place

on the page and getting TEX to make room for it. There are two basic approaches. The first is to use the automatic sizing described in section 2.1.1. The PostScript files that Illustrate produces follow the Adobe file structuring conventions. This first approach, however, is only implemented for plain TEX. Furthermore, the bounding box information that Illustrate provides is wrong. The second approach, sizing the illustration yourself, is described in this section.

There are several strategies for aligning the illustration and your paper. Strategy 1, for example, is to make the origin of the illustration the lower left corner and then use the `psbox` command defined above. That makes it tough to print the illustration out by itself because parts of the illustration are not in the imageable area of some printers. Another problem is that using the lower left corner makes centering the illustration more difficult (we must know its precise width and height). So instead we will use strategy 2 and align the center of the TEX box that surrounds the illustration with the center of Illustrate's page, which is at (4.25, 5.5).

Don't think too much about what that means because here's a simple procedure to follow. Make your drawing using Illustrate. When you are happy with it, center the illustration on the page using the Illustrate command `M-X Center Illustration`[5]. That command centers everything about the center of the page. Find the size of the illustration (in inches) by doing a mouse click on `Show Illustration Size`. Remember the size because we need to tell it to TEX. Then compile the illustration into a PostScript file.

In your TEX or LATEX file define the appropriate insertion macro; the definitions are given in section7. The resulting illustration is shown in figure 6. The illustration was inserted into this file using the commands

```
\begin{figure}
\begin{center}
\fbox{AB \illustratefile{2.0in}{1.0in}{/u/glr/illus1.ps} CD}
\end{center}
\caption{\label{illpic}Box, Letters, and Input from Illustrate}
\end{figure}
```

[5] If you don't like the way Illustrate centers the illustration, just draw a rectangle around the image that frames it the way you want. Make the rectangle color be white so it doesn't show up on the final print.

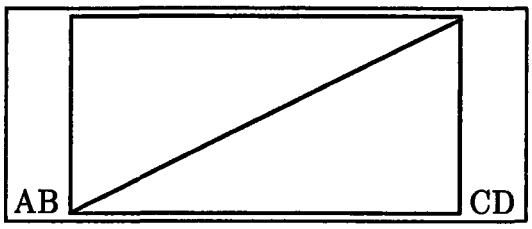


Figure 6: Box, Letters, and Input from Illustrate

The `fbox` command shows where TEX believes the box is. The additional text shows that the illustration baseline is at the bottom. Of course, you usually wouldn't include them in your document. The actual illustration is a boring 2 inch wide 1 inch high rectangle with one diagonal.

There are some problems with the `illustratefile` macro. It should not use the `psfile` translation mechanism because it is insensitive to the TEX magnification parameter. If we use a TEX translation scheme, then the user can specify conventional dimensions (subject to scaling) or true dimensions (no scaling). That should remove some of the trouble with magnification and work for YTEX.

### 2.1.3 MacDraw and MacPaint

Your are on your own here. The files generated by these programs do not stand by themselves. The file `tex.ps` (which is inserted by the Unix dvi converter) plays some games. You'll have to experiment.

MacDraw does not supply bounding box information in the header, so the automatic sizing method will not work. Someone commented that PostScript files made by MacDraw and MacPaint have their origin at the upper left corner of the page, but I really don't know.

### 2.1.4 Problems with the `psfile` Special

There are many problems with the `psfile` special command. The syntax of this special command is nonstandard; there should be a space after `psfile` instead of an equals sign. This syntactic problem is slight compared to the

unfortunate semantics of the command. And sadly, it seems that somebody had to work hard to make it do the wrong thing.

The special command should just insert a PostScript file using as the origin the absolute position of the special command. The TEX magnitude parameter should be in effect so that the entire document gets scaled (this is the meaning Knuth intended for this parameter). Unfortunately, the command undoes the existing magnification.

We should define two new special commands (so we don't conflict with existing files). `psraw` should interpret the rest of the special command as raw PostScript code (that should be interpreted under the current transformation). `psrawfile` should insert a file of raw PostScript code under the same conditions. The specification of dvi files allows special commands to be arbitrarily large, so we could have a TEX macro that inserts the illustration when TEX is run rather than when the dvi file is converted to the printer format. Many dvi converters, however, will choke on a special command 256 bytes long. The semantics of `psraw` are much cleaner because the dvi converter does not interpret a file name.

## 2.2 Merging QUIC

QUIC is not very convenient for merging illustrations because most of its positioning commands require absolute coordinates. We are also phasing out the QUIC printers. When you edit your TEX file, the figure locations probably move, and consequently you will have to regenerate the QUIC files using these new locations[6]. If you insist on using QUIC, then there are two easy ways out of this problem. Use full page illustrations because their position on the page will never change, or use the (small) subset of QUIC commands that allow relative positioning. Bitmaps, for example, require no absolute positioning commands[7].

There is also a problem with using fonts: you should use QUIC's internal (ROM) fonts or choose some random font number that does not clash with a font number that TEX uses. There are 32,000 font numbers available, so you can easily avoid a font clash, but be on the lookout for this problem.

[6]Furthermore, you will either have to read the dvi file or use a ruler to find the new positions.

[7]Illustrate output has absolute positioning commands in it.

When figures are inserted, the state of the QUIC processor is indeterminate. The inserted file should issue a QUIC `Isyntax` command to specify whether the coordinates are in pixels or mils. In addition some commands let you use a 4 bit or an 8 bit data mode; be sure the file explicitly sets the mode it uses.

The appropriate special commands are

```
\special{quicraw <quic commands>}
\special{quicrawfile <filename>}
```

QUIC uses a left handed coordinate system with the origin in the upper left corner of the page.

## 2.3 Image Format Files

If you are trying to get gray scale images into your text, then you must get them halftoned. While our printers have 300 pixel per inch resolution, using dithered (or error-propagated) images produces poor results. If the dithered pixels are small enough to make the picture look good, then the picture will not reproduce well. If you still believe in dithering your images, then you are crazy and should see BKPH for treatment. To halftone an image, use the `grok` or `halftone` programs that reside in `pig:[vision.utils]`. These programs convert gray scale (many bit per pixel) image files into one bit per pixel halftoned image files. The one bit per pixel image format files may just be inserted as long as the file is accessible to the machine running the dvi converter. See section 4.3 about file access.

The special command is

```
\special{image <zoom> <filename>}
```

The `zoom` parameter is optional and must be an integer. The origin is the lower left corner.

## 2.4 Mugshots

This special command only works on the LispM dvi converter, and is not recommended because the images are dithered instead of halftoned. If you want to use this facility, figure it out yourself. The special command is:

`\special{mugshot <name>}`

The origin is the lower left corner.

# 3 Printing on Big Sheets

If you want your output to come out on a larger than letter size piece of paper, you'll have to do two things. First, you have to get TEX and the dvi converter tuned in to what you want. Even with all the special command maneuvering below, the Unix dvi converter for PostScript won't do the right thing. You will have to run the dvi converter by hand (section 4.1). Second, when you actually print the file, you will have to do the paper tray shuffle; see section 5.

You get two basic choices of large paper size: 8.5x14 (legal) and 11x17. Though you may want your final output to be on 11x17 model paper, it may be less trouble to print each column on legal size paper and paste up the result. An additional advantage with printing each column on legal paper is that you can use the LaserWriter or PS800 for the final output; both of these printers have higher output quality than the LG1200, LG2400, or PS2400. The final conference proceedings, however, will be photoreduced, so the difference may not be discernible.

Two different processes must know that you are going to print on large paper. First, you must tell TEX to use large paper by setting the page margins. Second, you must tell the dvi converter about the larger page size so it can tell the printer to do the right thing. In theory, we would like to tell the dvi converter to use the large paper by inserting a special command into the TEX file, but in practice this only works for the Lisp Machine dvi converter and not for Unix.

You tell TEX about the larger paper by setting the page margins. The basic LATEX variables for setting the page size are `textheight` and `textwidth`. Margins are adjusted using `oddsidemargin`, `evensidemargin`, and `topmargin`. For TEX and YTEX, use the `hsize` and `vsize` variables, and adjusted the margins with `hoffset` and `voffset`[8]. Watch out for the built in one inch offset in both TEX and LATEX margins; setting the top margin to 0.5 inch actually produces a 1.5 inch margin.

[8] I don't know how `hoffset` is adjusted for odd-even page hacks.

```
\special{tray <n>}    % take paper from a particular tray
\special{letter}      % 8.5x11  paper (the default)
\special{note}        % 8.5x11  paper (hack for laserwriter)
\special{legal}       % 8.5x14  paper
\special{11x17}       %  11x17  paper
\special{landscape}   %  11x8.5 paper (landscape)
\special{landscape}   %  14x8.5 paper (may not work)
\special{ledger}      %  17x11  paper (landscape)
```

Figure 7: Special commands for paper size.

In theory, you tell the dvi converters (and the Lisp Machine dvi previewer) about the page size by using the TEX special commands show in figure 7. The processing of these commands by the different programs is inconsistent. Only include one of these special commands if it is needed (ie, don't put in a `letter` command). For the present, you should put the special command both at the beginning (see below for the placement of the command) and at the end of the file. The specials must be shipped out before the first mark is made on the page; for the special at the end of the file, just print a blank page that only has a special on it. If you get weird results printing to a PostScript printer, then take the commands out and run the dvi converter by hand (see section 4).

The `tray` command is required for printing to a QUIC printer because the controller is not smart enough to figure out the paper size. It is not needed for PostScript controllers if the paper size uniquely determines the tray; the only time you should need to use the command for a PostScript controller is if you want to print on some unconventional 8.5x11 paper such as letterhead.

The information in the special command must reach the processes before they have printed anything on the page. If something is printed before the page size is changed, then it will get positioned using the coordinate system for a conventional 8.5x11 page, and it probably will not end up where you want it. It's easy to tell TEX the paper size is, but it can be difficult to cause the special commands to come out before anything else on the page. To trick LATEX, not only do we have to attach the special command to the

```
% -*- Mode: TeX -*-
\hsize=5.0in
\vsize=12.0in
\special{tray 3}  %% Required for QUIC Printer
\special{legal}
  ... your text
\end
```

Figure 8: Printing on legal paper.

```
% -*- Mode: LaTeX -*-
\documentstyle[twocolumn,12pt]{article}
\textwidth   5.0in
\textheight 12.0in
\begin{document}
\title{{\special{11x17}}Warm Superconductors}
\author{I. C. B. Dun}
\maketitle
  ... your text
\end{document}
```

Figure 9: Two column output on large paper.

title (which is the first thing it prints on the page), we must also trick LaTeX into issuing a repositioning command immediately after the special[9]. Here are some prototypes for printing on large paper. Figure 8 is a plain TeX example using legal paper. For printing on really big sheets of paper, you probably also want two column output. Figure 9 shows a LaTeX example that does 2 column output on 11x17 paper.

[9]The special command actually "prints" somewhere else on the page because it is positioned using the letter coordinate system

## 3.1 Printing Drafts on Small Paper

Actually printing drafts at full size is a lot of trouble. It would be better to print drafts on 8.5x11 paper and only go to the trouble of printing on big paper for the final copy. The requirement would be that page breaks were the same in both situations. This nice idea is full of headaches, and you are probably better off using the dvi previewer and/or printing drafts on full size paper. The essential problem is that magnification does not work.

If you use the actual dimensions of the final output in your TEX source, then you won't run into as many problems with illustrations that don't like to be magnified. If you take this route, however, then you must reduce the output to fit it on an 8.5x11 sheet of paper. Our printing utilities, however, usually only understand magnifications greater than 1. When you print the file at mag = 833, the printer will substitute larger sizes for all of your fonts. The output will look goofy because all the characters will run together.

If you go the other way and use dimensions that when magnified turn out to be the actual dimensions, you'll be doing a lot of arithmetic. For lengths, the equation is easy; for setting the margins, the equations are more complicated because TEX's origin is not at a corner of the page.

How do you set the margins on the small sheet of paper so that when it gets printed out at magnification $m$ the margins are in the right place? Let $(x, y)$ be coordinates on the unmagnified (letter size) piece of paper and $(X, Y)$ be the coordinates of the enlarged print. All coordinates are relative to a lefthanded system with the origin in the upper left corner. Lengths are related by

$$l = L/m$$

The actual coordinates (inches) are related by

$$(x-1)m = (X-1)$$

$$(y-1)m = (Y-1)$$

So if you want the large paper to have a left margin to be 2.0 inches ($X = 2$), a top margin to be 0.75 inches($Y = 0.75$), and the line width to be 7.0 inches ($L = 7$) after magnifying it by 1.2, then set the unmagnified margins to be:

$$x = ((X-1)/m) + 1 = 1.8333333$$

$$y = ((Y-1)/m) + 1 = 0.7916667$$

$$l = L/m = 5.833333$$

The appropriate plain TEX commands are (notice we have to subtract the one inch from the margin values).

```
\mag=1200                  % only for final output
\special{11x17}            % only for final output
\hsize=5.83333in
\hoffset=0.83333in
\vsize=10.0in              % 12 inch final length
\voffset=-0.20833in        % 0.79167 - 1.0
```

When the magnified file was printed, the actual margins were 1.9 inches for the left margin and 0.83 inches for the top margin. The unmagnified margins were 1.82 inches and 0.80 inches when printed on le Monde, but were 1.84 inches and 1.29 inches when printed on Pravda! Pravda, it turned out, was way out of adjustment. To get your dimensions to come out right may involve tweaking the parameters in your TEX source or tweaking the local serviceman, so be on guard for bugs in unexpected places. The paper registration of the printers is supposed to be about 0.060 inches.

By the way, kosher values of magnification are 1.000, 1.095, 1.200, 1.440, and 1.728; these are progressions of 1.2 with a $\sqrt{1.2}$ thrown in for good measure[10]. Not all of these magnifications exist for all fonts. You should not have trouble with a magnification of 1.2, but most of our fonts don't have a magnification of 1.44. You're losing if your print out contains a error page that mumbles something about font substitutions.

# 4 The .dvi Converters

If you are going to merge illustrations that are not accessible to the printer host, if you are going to use big paper, or if your dvi file is too complicated to print in one piece, then you will have to explicitly run the dvi converter. If you are not doing one of these operations, then just print your dvi file as you always did.

On a Lisp Machine, give the `Hardcopy File` command. This command invokes the Lisp Machine dvi converter which can find files anywhere. It

[10]The real reason is to make an 11pt font for γTEX?

takes keyword arguments to specify the beginning and ending page. While this converter has the most complete coverage, it is also the slowest running and frequently causes the printer to run out of memory.

On OZ, giving the command `lpr foo.dvi` actually sends the file to Prep or Hermes and runs the dvi converter on that Unix host.

On a Unix, giving the command `lpr -d foo.dvi` does not run the dvi converter on your local machine unless the printer is also connected to your machine (ie, you are on Prep or Hermes). For example, if you are on the Sun named Wheaties, then giving the `lpr` command will send your dvi file to Prep, run `dvi2ps` on Prep, and then print the resulting PostScript file on Pravda. If you wish to insert some illustrations that are on the Sun (or reachable through NFS), then you must run dvi2ps locally.

The simplest way is to run the converter and pipe the result to the desired printer:

```
unix>dvi2ps  -r                                  file.dvi
       | lpr -v -P le-monde
unix>dvi2ps  -a /usr/lib/tex/pxlfonts.xerox file.dvi
       | lpr -v -P pravda
unix>dvi2qms -a /usr/lib/tex/pxlfonts.xerox file.dvi
       | lpr -v -P national-enquirer
```

Each of these three example commands should be typed on one line; there should be no carriage return after `file.dvi`. `dvi2ps` is used for PostScript printers, and `dvi2qms` is used for QUIC printers. The `-r` switch stacks the pages in reverse order; you should use for LaserWriters and PS800 printer unless you like shuffling paper. The `-a` switch tells the converter to use the fonts for the Xerox engine; if you don't use this switch, the fonts will look washed out. The `-v` switch to `lpr` says the file is the format the printer wants (either PostScript or QUIC). The `-P` switch specifies which printer gets the honor of trashing your printout.

Alternatively, if you want to keep the PostScript file around for a while, you should redirect standard output to a file.

```
unix>dvi2ps -r file.dvi > file.ps
unix>lpr -v -P le-Monde file.ps
```

## 4.1 Getting the Right Size Paper

This section is for PostScript printers only. To print dvi files on larger than letter size paper requires that you use a Lisp Machine or that you run the dvi converter manually on a Unix machine. For the Unix dvi converter, in addition to doing everything in the previous section, you must also supply a paper size option (`-o`) to `dvi2ps`. The appropriate incantations are:

```
unix>dvi2ps -r -o legal file.dvi | lpr -v -P le-monde
unix>dvi2ps -a /usr/lib/tex/pxlfonts.xerox -o 11x17 file.dvi
       | lpr -v -P pravda
```

Other options are `ledger`, `landscape`, and `manualfeed`.

More information can be found by incanting

```
unix>man dvi2ps
```

on any Unix machine.

## 4.2 Printing Problems

Sometimes the printer will get an error printing your file. There are many possible reasons for the error, but we will only consider two.

The first error is a stack overflow error; it is usually spurious. If you just send your file again, it will probably print. The problem usually happens if the printer has been idle for a long time, and might have something to do with what it does in its idle time. If the problem persists, try printing subsets of pages (see below).

The second error is a "VMError". This error means that the printer ran out of memory printing your document. If you are printing normal sized sheets on a LaserWriter, then try using the `note` special command. The command reduces the imageable area of the page to squeeze out a few more bytes of memory. Alternatively, you could send the file to a PS2400 printer (they have more memory on them). If that doesn't work, then you must run the dvi converter yourself and select subsets of pages. For `dvi2ps`, you can use the `-f` and `-t` to specify the "from" and "to" pages. You might also want the `-r` switch to get reverse stacking. For example, this command will print pages 100 to 150 on the printer le Monde.

```
unix>dvi2ps -r -f 100 -t 150 thesis.dvi | lpr -v -P le-monde
```

## 4.3 Understanding the Machine Issue

If you use a special command that has a filename, then you must understand which machine will process that filename. Special commands are not interpreted by TEX; they are interpreted by the dvi converter. "dvi" stands for "device independent". The dvi converter takes the device independent output of TEX and translates it to the device dependent format required by the printer you will use. That format, for our purposes, will either be PostScript or QUIC.

When the dvi converter is doing this translation, we have the opportunity to merge some other device dependent information into the device dependent output – a PostScript illustration, for example. The dvi converter is just a program, and for it merge the file you want, that file must be transparently accessible to the host running the dvi converter.

The different machines that the lab owns have varying capabilities, so the definition of transparently accessible varies from one class of machine to the next. We are only concerned with 3 classes of machines.

- Lisp Machines can access any file on any machine in the lab. If you use the Lisp Machine dvi converter, specify the filenames just as you Zwei. The dvi converter, however, is slow and has problems. A 20 page dvi file takes a long time to convert and may not print successfully. If you are printing to a LaserWriter, you can use the note special command (section 4.2).

- VAXes running Unix have only token access to files on other machines, so you are best off only specifying the filenames of local files. In addition, the `dvi2qms` converter can access files on OZ, PIG, and the ITS machines; specify the host with the machine name followed by a colon. The `dvi2ps` converter cannot access these other hosts.

- Suns running Unix can access files of any machine that understands the NFS network protocol. That effectively means you can access any file on any Sun in the lab. In addition, you can access files on Lisp Machines if you use a Unix translation of the Lisp Machine pathname. The file `b:>glr>foo.bar` is referred to as `/b/glr/foo.bar`; you may have to mount the Lisp Machine disks: become the super user and run `/etc/mount /b`.

| Special | LispM | dvi2ps | dvi2qms |
|---|---|---|---|
| image | x | | x |
| psfile | x | x | |
| psraw | | | |
| psrawfile | | | |
| tray | x | x | x |
| note | x | x | |
| letter | x | x | x |
| legal | x | x | x |
| ledger | x | x | |
| 11x17 | x | x | |
| quicraw | x | | x |
| quicrawfile | x | | x |
| mugshot | x | | |

Table 1: Recognized Special Commands

All of our printers are currently connected to VAX hosts that have limited file access to the rest of the world. The pictures in this document, for example, are stored on Prep and printed on Pravda or le Monde. To print the file I run LaTeX and then run `lpr -d papers.dvi` on Prep.

Another approach is to keep your files on a Sun such as Wheaties and then explicitly invoke the dvi converter.

## 4.4 Recognized TEX `Special` Commands

Table 1 lists the special commands and the dvi converters that understand them. Using commands that are not recognized may cause errors.

## 4.5 dvi Converter Source Files

The AI Laboratory's canonical sources for the Unix dvi converters are located on Wheaties (for Suns) in

```
/src/local/usr.lib/lpr/filters/qms
/src/local/usr.lib/lpr/filters/apple/dvi2ps/*
```

and on Hermes (for Vaxes) in

```
/usr/src/local/usr.lib/lpr/filters/qms
/usr/src/local/usr.lib/lpr/filters/apple/dvi2ps/*
```

If you edit these files, be keep the copies on Wheaties and Hermes consistent. When you install the updated code on Hermes, you must also copy (`rcp`) the binaries to the other Vax Unix hosts: Vulcan, Prep, Hermes, and HT. The Suns automatically get the new binaries from Wheaties.

The source for the PostScript prolog file, `/usr/lib/tex.ps`, is also in the `dvi2ps` directory; it should also be installed locally and on all Vaxes whenever it is changed.

Never edit the distributed copies; your changes will get wiped out the next time somebody does it right.

# 5 The Paper Tray Shuffle

Now that your dvi file has been converted and is about to be printed, you'll have to shuffle some paper trays to make it come out on the right size paper. Of course, if you are printing onto conventional letter size paper, then you don't have to worry about this at all.

The first order of business is to find out what print engine your output will be produced on. You can look this up in the tables at the end of this document, or you can figure it out by looking at the printer. If it's much bigger than a bread box, it's a Xerox engine. Otherwise it's a Canon engine. Another dead giveaway is the number of paper trays: the Canon engine only has one.

## 5.1 The Canon Engine

The Canon engine does not print 11x17 paper, so this section only addresses printing on legal size paper[11]. We would like the machine to be intelligent and tell you to change the paper tray when it needs legal or letter paper, but

[11] You can print on smaller sheets using the manual feed option (that is not described here).

it doesn't know how to do that yet[12]. We will pretend it does anyway, and these instructions describe how it should work.

First find the legal size paper tray (cassette) and queue your file for printing (the letter tray should be in the machine). When the printer needs legal paper, it will turn on a yellow light and give you 60 seconds to exchange the trays. All you have to do in that 60 seconds is pull the letter tray out and put the legal tray in. If you don't change the tray in time, the system aborts your print job.

When your job is finished printing, swap the trays back. The 60 second timeout also applies to the other jobs in the queue; if they don't have a letter tray in the machine, then the system sequentially cancels their jobs.

## 5.2 The Xerox Engine

Printing legal size paper on the Xerox engines is easy – just put the legal size paper in tray 3[13] (which is the lower of the two small trays). You can leave the printer set up that way until some one else needs to use the tray for something else (like printing on letterhead). When your job prints, it will just take paper from tray 3.

The Xerox Engine can also print on 11x17 paper, but only the PostScript controller knows how to do it. So to print on 11x17, you must not only use a Xerox Engine, but you must also use a PostScript printer[14].

You should follow these steps to set the printer up for 11x17 paper.

- Read the instructions about loading the large paper feeder in the printer manual – a copy is next to the machine.
- Turn the paper tray knob and replace colored paper in tray 2 with plain 8.5x11 paper. You need not stop the printer for this operation; it

[12]Currently the printer will print on any paper that is in the tray, so you must have the tray loaded before your job starts printing. The safest way to do this is to disable queueing for the printer (see the printer cheat sheet), change the paper tray, enable queueing long enough to submit your file. When your file is finished printing, put the letter tray back in and enable queueing.

[13]We have one Xerox engine that has only 2 small trays. You'll have to do a quick change maneuver on tray 2 to print legal size sheets on it. Just after the printer prints your cover sheet, turn the paper handle and swap the trays. When your job is done, swap the trays back.

[14]Currently only Pravda has these dual capabilities.

should continue to print out of the main paper tray while you change tray 2. The machine will now print white (instead of colored) cover sheets.

- When the printer is between printing jobs, load the 11x17 paper into the large feeder (the main tray). The controller is smart enough that it now knows the main tray now has large paper in it (a little microswitch notices the large paper door is open), so everybody else's stuff should get printed out of tray 1.
- Submit your print job. Your job will then feed out of the large paper tray. Any other print jobs in the queue should take paper out of tray 1.
- When you are finished printing, then unload the large paper from the main tray and restock it with letter size paper.
- When the printer is between printing jobs, turn the paper tray knob and put colored paper back into tray 2. When you're done, be sure you turn the paper tray knob back to vertical.

This procedure should work, but if other jobs are getting printed on large paper, you should disable the printer queue while you print your document. The printer cheat sheet tells how to disable the queue. You'll have to briefly enable the queue to submit your print job, but then immediately disable it. Disabling the queue prevents other people from using it, so you should not monopolize the printer for a long time.

## 5.3 Printer Characteristics

The AI Laboratory currently has several printers. Table 2 lists the printers we have as of 8 July 1987 and which floor they are on. The table also lists the printer's host – the machine that manages the printer queue. The host is also the machine that runs the dvi converter if you just spool the file to the printer (ie use `lpr` command without explicitly running the dvi converter). The last column of the table is the printer model, eg PS2400, which is used to lookup the printer's characteristics in the next table.

Table 3 lists the printer's language (PostScript or QUIC), the top speed of the printer, whether it can print on 11x17 inch paper, and which order

| Printer | Floor | Host | Man. | Model |
|---|---|---|---|---|
| Pravda | 7 | Prep | QMS | PS2400 |
| le Monde | 7 | Prep | Apple | LaserWriter |
| The Washington Post | 7 | Hermes | QMS | PS800 |
| Daily Planet | 8 | Hermes | QMS | LG2400 |
| The Wall Street Journal | 9 | Hermes | Apple | LaserWriter |
| National Enquirer | 9 | Hermes | QMS | LG1200 |
| Aime | 1 | HQVax? | QMS | PS1200 |
| Pulp | 2 | Nutmeg | QMS | PS800 |
| Ham | 3 | Nutmeg | QMS | PS2400 |
| Charmin | 3 | Charmin | Imagen | 8-300 |
| Salami | 4 | Nutmeg | QMS | PS800 |
| Oval | 4 | Oval | Imagen | 8-300 |
| GI | 5 | Nutmeg | QMS | PS2400 |

Table 2: Printer Names, Locations, Hosts, and Models

the pages are stacked (`-r` means back to front). LaserWriters, for example, use PostScript and don't print on 11x17 paper. All our printers can print on letter or legal size paper.

A final twist on the paper size issue is that only a portion of the page is available for printing. Table 4 describes the imageable areas for LaserWriters, PS800s, and PS2400 printers. The imageable areas are centered on the physical page.

# 6 Bibliography

# References

[1] *QUIC Programming Manual, Version 3.1.* Quality Micro Systems, Mobile, Alabama, 1985.

[2] Daniel Brotsky. *How to Use YTEX.* Working Paper 273, MIT Artificial Intelligence Laboratory, Cambridge, MA, October 1985.

| Model | Language | pages/ minute | print 11x17? | stacks | Print Engine |
|---|---|---|---|---|---|
| Laserwriter | PostS. | 8 | no | -r | Canon |
| PS800 | PostS. | 8 | no | -r | Canon |
| 8-300 | Impress | 8 | no | -r | Canon |
| LG1200 | QUIC | 12 | no | | Xerox |
| LG2400 | QUIC* | 24 | no | | Xerox |
| PS1200 | PostS | 12 | no | | Xerox |
| PS2400 | PostS | 24 | yes | | Xerox |

Table 3: Printer Characteristics

| paper size | dimensions | LaserWriter PS800 | PS2400 | units |
|---|---|---|---|---|
| letter | 8.50 x 11.0 | 8.00 x 10.92 | 7.950 x 10.92 | in |
| note | 8.50 x 11.0 | 7.69 x 10.16 | NA | in |
| legal | 8.50 x 14.0 | 6.72 x 13.00 | 8.000 x 13.50 | in |
| ledger | 17.0 x 11.0 | NA | 16.50 x 10.56 | in |
| 11x17 | 11.0 x 17.0 | NA | 10.56 x 16.50 | in |
| a4 | 210. x 297. | NA | 197.3 x 284.5 | mm |
| a3 | 297. x 420. | NA | 280.4 x 400.8 | mm |

Table 4: Imageable areas for the different printers

[3] Adobe Systems Incorporated. *PostScript Language Reference Manual.* Addison-Wesley, Reading, MA, 1985.

[4] Adobe Systems Incorporated. *PostScript Language Tutorial and Cookbook.* Addison-Wesley, Reading, MA, 1985.

[5] Donald E. Knuth. *The TEXbook.* Addison-Wesley, Reading, MA, 1984.

[6] Leslie Lamport. LaTeX*: A Document Prepartion System.* Addison-Wesley, Reading, MA, 1985.

# 7 The Macros

For reasons I do not understand, LaTeX won't understand macros that are written in plain TeXs. Thus there is the agony of writing two definitions for each macro: one for LaTeX and the other for the conventional TeX dialects.

The macros are also subject to change. These macros are sufficient to get by, but they also need to be fixed up. Sadly, there is not enough time to spend on that project. So here are the macro definitions for both TeX and LaTeX.

The source files for this document and its macros are located in the following files. You may want to look at the sources to check for changes.

| | |
|---|---|
| Text | `OZ:OZ:<GLR.TEXT>PAPERS.TEX` |
| LaTeX Macros | `OZ:OZ:<GLR.TEXT>PAPERL.TEX` |
| TeX Macros | `OZ:OZ:<GLR.TEXT>PAPERT.TEX` |
| PSAdobe Macros | `OZ:OZ:<GLR.TEXT>PSADOBE.TEX` |

## 7.1 Plain TeX Macros

The plain TeX version of the macros:

```
% -*- Mode: TeX -*-
%
%%%% Plain TeX Macros for Merging Illustrations
%
%%% Gerald Roylance 1987
%
\def\psbox#1#2#3{%
```

```
    \hbox to #1{\vbox to #2{\vfil}\special{psfile=#3}\hfil}}
%
% 8.5/2 = 4.25in = 306 PostScript Points
%  11/2 = 5.50in = 396 PostScript Points
%
\def\illustratefile#1#2#3{%
   \hbox to #1{%
     \hfil%
     \vbox to #2{%
       \vfil%
       \special{psfile=#3 hoffset=-306 voffset=-396}%
       \vfil}%
     \hfil}}
\endinput
```

## 7.2 LaTeX Macros

The LaTeX version of the macros:

```
% -*- Mode: LaTeX -*-
%
%%%% Plain TeX Macros for Merging Illustrations
%
%%% Gerald Roylance 1987
%
\newcommand{\psbox}[3]{%
   \makebox[#1][l]{%
      \parbox[b]{0.0in}{\vspace*{#2}}%
      \special{psfile=#3}}}
%
% 8.5/2 = 4.25in = 306 PostScript Points
%  11/2 = 5.50in = 396 PostScript Points
%
\newlength{\illheight}
\newcommand{\illustratefile}[3]{%
  \makebox[#1]{%
    \setlength{\illheight}{#2}% So we can divide
    \raisebox{0.5\illheight}[\illheight][0.0in]{%
```

```
        \special{psfile=#3 hoffset=-306 voffset=-396}}}}
\endinput
```

## 7.3 PSAdobe Macros

These macros are for inserting PostScript files that follow the Adobe file structuring conventions and include the bounding box information in the header. These macros do not work for LaTeX.

```
% -*- Mode: TeX -*-
%
%%% Using the Adobe PostScript File Structuring Conventions
%
%%% Gerald Roylance 1987
%
%%%% Read a PostScript Header
%
%%% The header looks like
%%%
%%%     %!PS-Adobe-1.0
%%%     %%Creator: Illustrate Version 11.0
%%%     %%CreationDate: 6/23/87 17:35:01
%%%     %%Pages: 1
%%%     %%BoundingBox: 466.12912 718.63934 612.0283 792.0283
%%%     %%EndComments
%%%
%%% *** Should complain if #2 is (atend)
%
%%% Dimensions of the illustration
%
\newdimen\psadobex
\newdimen\psadobey
\newdimen\psadobew
\newdimen\psadobeh
%
%%% Use the TeX calling mechanism to parse the header.
%%% -- assumes the picture will be scaled
%%% -- Result is left in some global registers
```

```
%
\def\psadobeheaderparse !PS-Adobe-#1BoundingBox: #2 #3 #4 #5
   {\global\psadobex= #2 bp
    \global\psadobey= #3 bp
    \global\psadobew= #4 bp
    \global\advance\psadobew by -\psadobex
    \global\psadobeh= #5 bp
    \global\advance\psadobeh by -\psadobey
    \endinput}
%
%%% Read the Adobe header
%
\def\psadobeheader#1{%
  \begingroup
     \catcode`\%=9  % ignore comment character
     \expandafter\psadobeheaderparse\input #1
     \endgroup}
%
%%%% Automatic Sizing of a PostScript File
%
% \mag is an integer, but PostScript wants a float.
% This gross hack works if \mag < 10000
% Sorry, but \mag is converted when this file is read in;
% it didn't work to do it on the fly -- I'm not a TeXhacker.
%
\newcount\float \float=\mag \advance\float by 10000
\def\floater#1#2#3#4#5{#2.#3#4#5}
\def\floatmag{\expandafter\floater\the\float}
%
%%% Insert a file with scaling by \mag
%
\def\psadobe#1{%
  \psadobeheader{#1}%
  \psfile{\psadobex}{\psadobey}{\psadobew}%
         {\psadobeh}{\floatmag}{#1}}
%
%%% Insert a file without scaling by \mag
```

```
%%% convert scaled dimensions to true dimensions
%
\def\psadobenomag#1{%
  \psadobeheader{#1}%
  \multiply\psadobex by 1000 \divide\psadobex by \mag
  \multiply\psadobey by 1000 \divide\psadobey by \mag
  \multiply\psadobew by 1000 \divide\psadobew by \mag
  \multiply\psadobeh by 1000 \divide\psadobeh by \mag
  \psfile{\psadobex}{\psadobey}%
         {\psadobew}{\psadobeh}{1.000}{#1}}
%
% psfile(x,y,w,h,m,file)
%
\def\psfile#1#2#3#4#5#6{%
  \hbox to #3{%
     \kern-#1%
     \vbox to #4{%
        \vfil%
        \kern#2%
        \special{psfile=#6 hscale=#5 vscale=#5}%
        \kern-#2}%
     \kern#1\hfil}}
%
\endinput
```