1 Output Specification

This section defines the output language which is generated by TPIC. It is intended only for use by writers of DVI device drivers. The first lines of output will execute a `\newbox` command to define `\graph` if it has not already been defined. Each picture is enclosed in a `\vtop` which is assigned to `\graph`. The width and height of this box are the size of the picture as known to TPIC (text doesn't count because TPIC can't know how wide it may be). Inside this box are a sequence of `\special` commands which define all the graphics, along with `\hbox` commands setting the text left, right, or center of a given point.

Each graphical element is placed relative to TEX's idea of the location of this `\special` on the page. Each `\special` is set at the top left corner of the `\graph` box. This point is considered the origin of the graph. The x axis is positive to the right, and the y axis is positive moving down the page. Thus the coordinates in the graphics commands are in terms of a virtual page with axes oriented the same as on many printers and workstation screens, like this:

```
.PS
"0,0" at (0,0) above
arrow from (0,0) to (0.5,0)
"\ $+x$" at (0.5,0) ljust
arrow from (0,0) to (0,-0.5)
"$+y$" at end of last arrow below
.PE
\centerline{\box\graph}
```

Angles are measured in the conventional way, from $+x$ towards $+y$. Unfortunately, that reverses the meaning of "counterclockwise" from what you see in the output. Angle measurements are in radians, from 0 to 2π. A complete circle or ellipse will have *start angle* $= 0$ and *end angle* $\geq 2\pi$. Thus, each distance (expressed in milli-inches) will be a positive number, since the box encloses all of the graphical elements.

The possible `\special` commands which must be implemented are as follows. All numbers which represent distances are integers except where noted. Angles are always measured in radians, and they are measured in the conventional way from $+x$ towards $+y$. Normally, this would be "counterclockwise".

Unfortunately, since the y axis points down instead of up, this turns out to be backwards. This is especially confusing because in the input language, $+y$ point upwards. Device driver writers should beware of this point. It is best to do all arithmetic in the coordinates of TPIC's output, then convert the resulting endpoints to the device's coordinates for printing; this should produce correct results in all cases.

pn s Sets the pen size to s milli-inches.

pa x y Add point (x, y) to a "path" which is being stored in the DVI interpreter. The values of x and y are in milli-inches.

fp Flush (i.e., print using the current pen size) a previously defined path. The number of path elements is reset to zero. If shading has been specified, and if the path is closed, then the enclosed area should be shaded. Currently, TPIC can only specify closed regions which are rectangles using the `pa` command, and circles/ellipses are specified with the `ar` command; however, device driver writers should not rely on this restriction. Someday, TPIC may be able to specify arbitrary closed areas using `pa` commands. Currently, the only example of this is the shading of arrowheads, which are triangular.

ip This is the same as `fp` except that the path is not actually drawn. Shading of the defined region is performed if it has been specified.

da f This is the same as `fp` except that the line is drawn as a dashed line with f inches per dash (f is a real number).

dt f This is the same as `da` except that a dotted line is drawn, with f inches between each dot (f is a real number).

sp d This is the same as `fp` except that a spline is drawn through the points. The spline is guaranteed to go through only the first and last points, depending on the relative positions of any intermediate points. The argument d is an optional real value. If $d = 0$ or the argument is omitted, the spline is drawn normally, as a solid line. If $d > 0$, it is dashed with d as the dashwid, else if $d < 0$, the spline is dotted with $-d$ as the dotwid.

ar x y r_x r_y s e Draw an arc with center at (x, y), from starting angle s to ending angle e. If it's a complete circle or ellipse, then r_x and r_y determine the x and y radii, respectively. Otherwise, $r_x = r_y$ will hold, and an arc from s to e should be drawn.

ia x y r_x r_y s e The `ia` command is to `ar` what `ip` is to `fp`.

The following command is optional in DVI driver implementations of TPIC. Drivers which do not implement it should silently ignore it.

sh s Shade the interior of the next closed figure defined with three or more `pa` commands followed by a `fp` or `ip` command, or by an `ar` or `ia` command. The value s is a floating point number from 0 to 1. Zero implies a completely white interior (including erasing anything underneath), while 1 implies completely black. A value of 0.5 is a default "normal" gray shading. If s is not specified, a value of 0.5 should be used. NB: Shading applies to the interior of the object only; the borders of the object are still drawn using the current pen, which is particularly important for dvi drivers which do not support the `sh` command. The pixels generated by the shading value should be "ANDed" with the pixels already in the area. Subsequent text pixels should be "ORed". Thus, text *followed* by a "`sh 0`" command covering the same area will disappear. Any shading value greater than zero should add the shading to the existing text, with the text remaining visible. Of course, a shading value of one will be completely black, so any text that is there won't be distinguishable.

The following two opcodes are considered obsolete, but they probably should be implemented for backwards compatibility:

wh This is the same as "`sh 0`".

bk This is the same as "`sh 1`".

tx This command was used to specify a texture to be used in shading instead of the default one. Its use was limited to only one particular Imagen device driver, and it was very device-specific. Its implementation is discouraged in favor of support for the argument to the `sh` command.