

Dymo printers from Perl

by brian d foy
brian.d.foy@gmail.com



Part of my job as the publisher of The Perl Review is to send people their copies of the magazine. When a new issue comes out, the bulk mail is easy, happens all at once, and is done. However, between issues, I need to send out a lot of mail too—a lot more mail than I ever thought I'd have to send. Not only do I have to send a lot of mail, but I need to keep track of what I've already sent.

For a long time, I wrote the address on the envelopes when I had to send out single copies, and a bit after that became tedious, I finally bought myself a label printer, the Dymo Labelwriter 400 that you see on the cover. As with any new hardware, I want to control it programmatically from Perl.

To put it all together, I had to extract some information from the Dymo printer description files, modify the `PostScript::MailLabels` module, and integrate it with the *TPR* subscriber database.

■ The old way-----

Before my Dymo solution, I was already using `PostScript::MailLabels`. The primary use of the module is to print labels on multi-label sheets that are close to a standard Avery label format. This works very nicely when I need to print hundreds of labels for bulk mailing, but not so well when I want a single label.

Despite that, `PostScript::MailingLabels` can handle single labels because it's smart enough to know how many labels are on a sheet and can start printing on any one of them. I can easily make a tiny program that I can use to print a single label:

```
use PostScript::MailLabels;

my $labels = PostScript::MailLabels->new;

my $setup = $labels->labelsetup(
    # various set up outputs
    Avery      => 8320,
    FirstLabel => $ARGV[0],
);

my $addr = # maybe some extra processing
    map { my $s = chomp; $s } <STDIN>;

my $output = $labels->makelabels( $addr );
```

```
# do something with the output
```

The string in `$output` that I get back from `makelabels` is PostScript. I can send it directly to a PostScript printer, save it in a file, or whatever else I want to do with it.

Outside of the Perl portion, however, printing a single label means I have to look at my available sheet of labels, figure out which label is the next one, the correctly load the label sheet into the printer. Guess which one of those I always got wrong. Yeah, most of them. It didn't feel like an automated solution when I had to do so much work myself.

■ PostScript::MailLabels data-----

The `PostScript::MailLabels` is a very nice module that allows me to deal with most of the problems of putting several lines of text on a label. It knows how to deal with margins, text sizes, fonts, and other features that I don't want to think about re-implementing. I want to keep using it, but I need to make it Dymo-aware.

`PostScript::MailLabels` already knows about Avery labels by their product codes that it stores in `PostScript::MailLabels::BasicData`. With very minor adjustments to the rest of the module, which I won't show in this article, I can make the module work with my new label printer by adding its label data next to the Avery data. *Listing 1 (next page)* shows an extract of the basic data which I'll need to follow for the Dymo data.

Each pre-defined label type in has several things it knows about the output:

- label type
- paper size
- number of labels per sheet
- the size of the label
- the printable area of the label

I need to collect this information for the Dymo label that I want to use, a single address label measuring 1.125 x 3.5 inches, has the product code 30252. That's not a particularly interesting number, but I figure that I might want to use a different label later. The 30252 is big enough to fit a reasonable address.

However, it often doesn't fit an address for some countries, such as the United Kingdom, where some addresses need up to eight lines including the recipient's name. Since I might want to use different labels later, I figure I should add data for all