Printing Envelopes and Labels in LaTeX 2$_\varepsilon$: *EnvLab* Package *†

Boris Veytsman

1997/07/16

Contents

*This file has version number v1.2, last revised 1997/07/16.

1 Introduction

The standard `\makelabels` command in the LaTeX 2$_\varepsilon$ `letter.cls` documentclass typesets labels on Avery 5352 sheets. A typical user may want more. *EnvLab* redefines `\makelabels` in[1] a more useful and customizable way

The detailed usage of the package is described in the file `elguide.tex`. Here we just comment the macros.

2 Identification

First, we must say "Hello world."

```
1 ⟨*package⟩
2 \NeedsTeXFormat{LaTeX2e}
```

`\envlab@ok` `\envlab@oops` Now let us check whether we in the letter documentclass. Actually we will accept any class that has `\makelabels` defined (custom letter classes, etc.)

```
3 \def\envlab@oops{%
4   \PackageError{envlab}%
5   {Envlab is used outside of \MessageBreak%
6    a letter-compatible documentclass}%
7   {You are trying to use Envelopes & Labels\MessageBreak%
8    package, but your documentclass does not\MessageBreak%
9    understand address formatting commands.\MessageBreak%
10   Try standard document class letter\MessageBreak}}
```

[1] hopefully

Media	Number per page	Rotation	Return address
Envelopes	One	Settable	Yes
Labels	Several	Not rotated	No
Big Labels	Several	Not rotated	Yes

Table 1: Differences between envelopes, labels and big labels

```
\def\envlab@ok{%
  \PackageInfo{envlab}%
  {Envelopes & Labels package: found makelabels...\MessageBreak%
  Seems everything is OK. Good luck.}}
\@ifundefined{makelabels}{\envlab@oops}{\envlab@ok}
```

3 Preliminary code

3.1 Switches, etc.

`\if@envelope` `\if@biglabel`

There are three kinds of things we can print: envelopes (default) labels and big labels The differences are summarized in Table 1.

```
\newif\if@envelope
\@envelopetrue
\newif\if@biglabel
\@biglabelfalse
```

`\if@rotateenvelopes` `\if@printreturnaddress`

Now we must determine whether we want to rotate envelopes and whether to include return address (both yes by default).

```
\newif\if@rotateenvelopes
\@rotateenvelopestrue
\newif\if@printreturnaddress
\@printreturnaddresstrue
```

`\@envelopeposition`

Now let us decide how to print envelopes. They can be either centered (default) *or* shifted to the left or to the right of the paper tray. The counter `\@envelopeposition` can be, correspondingly, either 0 or 1 or 2. The value of 3 corresponds to the "custom placing", when the user sets `\EnvelopeLeftMargin` manually.

```
\newcount\@envelopeposition
\@envelopeposition=0\relax
```

`\if@pswait` `\if@psautotray` `\PSEnvelopeTray`

The switches `\if@pswait` and `\if@psautotray` control optional manual feeding of envelopes and labels in *Postscript* printers (see Section 7.1 for details). The register `\PSEnvelopeTray` contains the name of the required tray.

```
\newif\if@pswait
\@pswaitfalse
\newif\if@psautotray
\@psautotrayfalse
```

```
\newtoks\PSEnvelopeTray
\PSEnvelopeTray={/otherenvelopetray }
```

`\if@barcodes` `\if@alwaysbarcodes` We can either print bar codes (default) or not. The second switch forces to print barcodes even if they are not last in the address (like `Pa 16801\\USA`)

```
\newif\if@barcodes
\newif\if@alwaysbarcodes
\@barcodestrue
\@alwaysbarcodesfalse
```

`\if@EL@redefine@opening` Now let us decide whether to mess with the `\opening` command.

```
\newif\if@EL@redefine@opening
\@EL@redefine@openingfalse
```

`\if@capitalizeaddress` Also, we can either capitalize the address (default) or not.

```
\newif\if@capitalizeaddress
\@capitalizeaddresstrue
```

3.2 Lengths and numbers

We want all lengths to be user settable, so no `@` in the names.

`\EnvelopeWidth` `\EnvelopeHeight` `\EnvelopeTopMargin` `\EnvelopeLeftMargin` An envelope has four basic lengths. The first two are self-evident. The third is the distance between the edge of the paper and the leading edge of the envelope. All pre-defined envelope sizes set this to zero. The fourth is the distance between the left edge of the paper and the envelope. Its value depends on the value of the `\@envelopeposition` variable. We will preset it to zero

```
\newlength{\EnvelopeWidth}
\newlength{\EnvelopeHeight}
\newlength{\EnvelopeTopMargin}
\newlength{\EnvelopeLeftMargin}
\setlength{\EnvelopeLeftMargin}{0pt}
```

`\LabelWidth` `\LabelHeight` `\LabelTopMargin` `\LabelLeftMargin` `\LabelRightMargin` A label has more parameters. The first two are the same as for the envelopes. The next two define the distances from the paper edges to the beginning of the labels. The last one describes the distance between the labels.

```
\newlength{\LabelWidth}
\newlength{\LabelHeight}
\newlength{\LabelTopMargin}
\newlength{\LabelLeftMargin}
\newlength{\LabelRightMargin}
```

`\c@LabelMaxCol` `\c@LabelMaxRow` The following numbers define, how many labels are in each row and how many rows are on each page

```
\newcounter{LabelMaxCol}
\newcounter{LabelMaxRow}
```

The lengths above are *external* parameters that determine an envelope or a label. Now we will describe *internal* lengths. We define them here because the commands `\SetEnvelope` and `\SetLabel` determine them basing on the given envelope or label type.

\FromAddressTopMargin
\FromAddressLeftMargin
\FromAddressHeight
\FromAddressWidth
\ToAddressTopMargin
\ToAddressLeftMargin
\ToAddressWidth

The following lengths are self-evident.

```
\newlength{\FromAddressTopMargin}
\newlength{\FromAddressLeftMargin}
\newlength{\FromAddressHeight}
\newlength{\FromAddressWidth}
\newlength{\ToAddressTopMargin}
\newlength{\ToAddressLeftMargin}
\newlength{\ToAddressWidth}
```

3.3 Main setting commands

OK, we are ready to set up envelopes and labels.

\SetEnvelope

The command \SetEnvelope has three parameters: optional top Margin, width and height of the envelope.

```
\DeclareRobustCommand{\SetEnvelope}[3][0pt]{%
  \@envelopetrue%
  \@biglabelfalse%
  \setlength{\EnvelopeTopMargin}{#1}%
  \setlength{\EnvelopeWidth}{#2}%
  \setlength{\EnvelopeHeight}{#3}%
  \setlength{\FromAddressTopMargin}{0.5in}%
  \setlength{\FromAddressLeftMargin}{0.5in}%
  \setlength{\FromAddressHeight}{0.33\EnvelopeHeight}%
  \setlength{\FromAddressWidth}{0.5\EnvelopeWidth}%
  \setlength{\ToAddressTopMargin}{0.5in}%
  \setlength{\ToAddressLeftMargin}{0.5in}%
  \setlength{\ToAddressWidth}{3in}}
```

\SetLabel

The command \SetLabel has seven parameters: five lengths and two numbers. All are mandatory:

```
\DeclareRobustCommand{\SetLabel}[7]{%
  \@envelopefalse%
  \@biglabelfalse%
  \setlength{\LabelWidth}{#1}%
  \setlength{\LabelHeight}{#2}%
  \setlength{\LabelTopMargin}{#3}%
  \setlength{\LabelLeftMargin}{#4}%
  \setlength{\LabelRightMargin}{#5}%
  \setcounter{LabelMaxCol}{#6}%
  \setcounter{LabelMaxRow}{#7}%
  \setlength{\ToAddressTopMargin}{0.1in}%
  \setlength{\ToAddressLeftMargin}{0.2in}%
  \setlength{\ToAddressWidth}{\LabelWidth}%
  \addtolength{\ToAddressWidth}{-\ToAddressLeftMargin}%
  \addtolength{\ToAddressWidth}{-\LabelRightMargin}}
```

\SetBigLabel

The command \SetBigLabel has seven parameters: five lengths and two numbers. All are mandatory:

```
\DeclareRobustCommand{\SetBigLabel}[7]{%
  \@envelopefalse%
  \@biglabeltrue%
  \setlength{\LabelWidth}{#1}%
  \setlength{\LabelHeight}{#2}%
  \setlength{\LabelTopMargin}{#3}%
  \setlength{\LabelLeftMargin}{#4}%
  \setlength{\LabelRightMargin}{#5}%
  \setcounter{LabelMaxCol}{#6}%
  \setcounter{LabelMaxRow}{#7}%
  \setlength{\FromAddressTopMargin}{0.0in}%
  \setlength{\FromAddressLeftMargin}{0.5in}%
  \setlength{\FromAddressHeight}{0.33\LabelHeight}%
  \setlength{\ToAddressTopMargin}{0.1in}%
  \setlength{\ToAddressLeftMargin}{0.5in}%
  \setlength{\ToAddressWidth}{\LabelWidth}%
  \addtolength{\ToAddressWidth}{-\ToAddressLeftMargin}%
  \addtolength{\ToAddressWidth}{-\LabelRightMargin}%
  \setlength{\FromAddressWidth}{\ToAddressWidth}}
```

4 Defining options

4.1 Envelope Sizes

```
\DeclareOption{businessenvelope}{\SetEnvelope{9.5in}{4.125in}%
  \PSEnvelopeTray={/com10envelopetray }}
\DeclareOption{executiveenvelope}{\SetEnvelope{7.5in}{3.875in}%
  \PSEnvelopeTray={/monarcenvelopetray }}
\DeclareOption{bookletenvelope}{\SetEnvelope{10.5in}{7.5in}}
\DeclareOption{personalenvelope}{\SetEnvelope{6.5in}{3.625in}}
\DeclareOption{c6envelope}{\SetEnvelope{162mm}{114mm}}
\DeclareOption{c65envelope}{\SetEnvelope{224mm}{114mm}}
\DeclareOption{c5envelope}{\SetEnvelope{229mm}{162mm}%
  \PSEnvelopeTray={/162x229cenvelopetray }}
\DeclareOption{dlenvelope}{\SetEnvelope{220mm}{110mm}%
  \PSEnvelopeTray={/dlenvelopetray }}
```

4.2 Labels sizes

```
\DeclareOption{avery5160label}{%
  \SetLabel{2.75in}{1in}{0.5in}{0.19in}{0.12in}{3}{10}}
\DeclareOption{avery5161label}{%
  \SetLabel{4.19in}{1in}{0.5in}{0.16in}{0.19in}{2}{10}}
\DeclareOption{avery5162label}{%
  \SetLabel{4.19in}{1.33in}{0.83in}{0.16in}{0.19in}{2}{7}}
\DeclareOption{avery5163label}{%
  \SetLabel{4.19in}{2in}{0.5in}{0.16in}{0.19in}{2}{5}}
\DeclareOption{avery5164label}{%
  \SetLabel{4.19in}{3.33in}{0.5in}{0.16in}{0.19in}{2}{3}}
\DeclareOption{herma4625label}{%
```

```
  \SetLabel{105mm}{42.3mm}{0mm}{5mm}{5mm}{2}{7}}
\DeclareOption{avery5262label}{%
  \SetLabel{110mm}{34mm}{21mm}{4mm}{5mm}{2}{7}}
\DeclareOption{avery5163biglabel}{%
  \SetBigLabel{4.19in}{2in}{0.5in}{0.16in}{0.19in}{2}{5}%
  \setlength{\ToAddressTopMargin}{0.1in}}%
\DeclareOption{avery5164biglabel}{%
  \SetBigLabel{4.19in}{3.33in}{0.5in}{0.16in}{0.19in}{2}{3}}%
```

4.3 Optional switches

All this should be evident...

```
\DeclareOption{rotateenvelopes}{\@rotateenvelopestrue}
\DeclareOption{norotateenvelopes}{\@rotateenvelopesfalse}
\DeclareOption{centerenvelopes}{\@envelopeposition=0\relax}
\DeclareOption{leftenvelopes}{\@envelopeposition=1\relax}
\DeclareOption{rightenvelopes}{\@envelopeposition=2\relax}
\DeclareOption{customenvelopes}{\@envelopeposition=3\relax}
\DeclareOption{printbarcodes}{\@barcodestrue}
\DeclareOption{noprintbarcodes}{\@barcodesfalse\@alwaysbarcodesfalse}
\DeclareOption{alwaysbarcodes}{\@alwaysbarcodestrue\@barcodestrue}
\DeclareOption{noalwaysbarcodes}{\@alwaysbarcodesfalse}
\DeclareOption{capaddress}{\@capitalizeaddresstrue}
\DeclareOption{nocapaddress}{\@capitalizeaddressfalse}
\DeclareOption{printreturnaddress}{\@printreturnaddresstrue}
\DeclareOption{noprintreturnaddress}{\@printreturnaddressfalse}
\DeclareOption{pswait}{\@pswaittrue\@psautotrayfalse}
\DeclareOption{nopswait}{\@pswaitfalse}
\DeclareOption{psautotray}{\@psautotraytrue\@pswaitfalse}
\DeclareOption{nopsautotray}{\@psautotrayfalse}
\DeclareOption{re}{\@EL@redefine@openingtrue}
\DeclareOption{nore}{\@EL@redefine@openingfalse}
```

4.4 Unknown options

All options we did not declare above are probably the options for the `graphics` package; let us send them there.

```
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{graphics}}
```

4.5 Default options

```
\ExecuteOptions{businessenvelope,rotateenvelopes,centerenvelopes}
\ExecuteOptions{printbarcodes,capaddress}
\ExecuteOptions{nopswait,printreturnaddress,nopsautotray,nore}
```

5 Configuration file

At this point we will look for the configuration file. This file is named `envlab.cfg`. The options declared in this file will supersede the ones declared in Section 4.5,

but will be in their turn be superseded by the options explicitly defined when the package is loaded.

```
\InputIfFileExists{envlab.cfg}{%
  \typeout{Loading configuration file envlab.cfg}}{%
  \typeout{Configuration file envlab.cfg is not found}}
```

Now let us discuss the structure of the configuration file. We want it to contain default options, and therefore to be loaded before the `\ProcessOptions` command. On the other hand we want it to contain some definitions that *supersede* the ones in this package. The hook `\AtEndOfPackage` helps to do this: we will just delay all definitions until later.

OK, let's construct an example of `envlab.cfg` file. The commands here essentially repeat Section 4.5, but we want to be foolproof...

```
⟨/package⟩
⟨*cfg⟩
%%
%% The default options go here
%%
\ExecuteOptions{businessenvelope,rotateenvelopes,centerenvelopes}
\ExecuteOptions{printbarcodes,capaddress}
\ExecuteOptions{nopswait,printreturnaddress,nopsautotray,nore}
%%
\AtEndOfPackage{\relax % Customization goes here
}
⟨/cfg⟩
⟨*package⟩
```

6 Processing options and loading packages

```
\ProcessOptions
\IfFileExists{graphics.sty}{%
  \RequirePackage{graphics}}{%
  \PackageWarning{envlab}{%
    You don't have the graphics package!\MessageBreak
    Probably you will not be able to print\MessageBreak
    envelopes sidewise. \MessageBreak}}
```

7 Document layout

7.1 Printer specific commands

`\@beginlabelshook` The command `\@beginlabelshook` is called at the beginning of the printing of envelopes and labels. We define it to be a no-op, but it is possible to introduce some printer-specific commands (like paper change).

```
\def\@beginlabelshook{\relax}
```

`\@beginlabelpagehook` The command `\@beginlabelpagehook` is like `\@beginlabelshook`, but is called at the beginning of each *page* of envelopes and labels.

```
\def\@beginlabelpagehook{\relax}
```

\AtBeginLabels
\AtBeginLabelPage

The hooks \AtBeginLabels and \AtBeginLabelPage redefine \@beginlabelshook and \@beginlabelpagehook. They are built like the standard LaTeX 2_ε hooks \AtBeginDocument, \AtBeginDvi, etc. In the implementation we use the *internal* LaTeX 2_ε command \g@addto@macro. In the current (June 1996) LaTeX 2_ε release it is defined as:

\g@addto@macro

```
\long\def\g@addto@macro#1#2{{%
  \toks@\expandafter{#1#2}%
  \xdef#1{\the\toks@}}}
```

We quote this definition in case *They* change Their minds...

```
\def\AtBeginLabels{\g@addto@macro\@beginlabelshook}
\def\AtBeginLabelPage{\g@addto@macro\@beginlabelpagehook}
```

\PSwait

PostScript printers can be switched to the manual feeding mode with the following code by *William Slough* <cfwas@eiu.edu>.

```
\def\PSwait{\special{ps: clear grestore @manualfeed 0 0 bop}}
```

The author explains his code in the following way:

> Here is a possible explanation:
>
> The clear removes operands from the PostScript stack, which has the effect of reversing some actions from the *previous* bop. Unfortunately, it reverses other important actions too (such as font size), but the grestore seems to get these back. Then the desired @manualfeed, followed by the "bop" for the beginning of page. The pair of 0's are used for bop and are completely bogus. However, from what I could detect, 0's are as good as anything. The values that DVIPS provides for bop seem to be related to DVI page numbers.
>
> I make no guarantee about the reliability of this solution, but initial tests indicate it will work for my environment.

\PSautotray

This implements the code by *Uri Blumenthal* <uri@.ibm.net>.

```
\edef\PSautotray{%
  \special{ps:clear grestore
    statusdict begin false setduplexmode
    /manualfeed true def
    \the\PSEnvelopeTray end 0 0 bop }}
```

The option pswait puts the \PSwait code in the beginning of each page. The option \psautotray puts there the \PSautotray code.

```
\if@pswait
  \AtBeginLabels{\PSwait}%
\else
  \if@psautotray
    \AtBeginLabels{\PSautotray}%
  \fi
\fi
```

7.2 Some useful counters for labels

\c@LabelCountCol
\c@LabelCountRow

These counters store the position of the currently printed label:

```
\newcounter{LabelCountCol}
\newcounter{LabelCountRow}
```

\c@LabelOffsetCol
\c@LabelOffsetRow

And these counters provide the offset for the label printed on a partially used sheet:

```
\newcounter{LabelOffsetCol}
\newcounter{LabelOffsetRow}
\setcounter{LabelOffsetCol}{1}
\setcounter{LabelOffsetRow}{1}
```

\FirstLabel

The command `\FirstLabel{`⟨*Row*⟩`}{`⟨*Col*⟩`}` sets the counters `LabelOffsetRow` and `LabelOffsetCol`.

```
\DeclareRobustCommand{\FirstLabel}[2]{%
  \setcounter{LabelOffsetRow}{#1}%
  \setcounter{LabelOffsetCol}{#2}}
```

7.3 Fonts

\@toaddressfont
\@fromaddressfont

We want the address to be printed in 12pt sans serif font. The return address will be printed in 10pt normal font.

```
\def\@toaddressfont{%
  \ifcase\@ptsize \large\or\normalsize\or\small\fi%
  \sffamily\selectfont}
\def\@fromaddressfont{%
  \ifcase\@ptsize \normalsize\or\small\or\footnotesize\fi%
  \normalfont}
```

7.4 Return address

\returnaddress

The standard letter class defines `\returnaddress` to be null. This is sensible if we are printing labels, but not so good if we are printing *envelopes*. Therefore let us redefine it:

```
\def\returnaddress{\fromaddress}
```

7.5 Margins, page styles, etc.

\startlabels

The command `\startlabels` is the internal command that prepares the paper for labels or envelopes, resets the internal counters and calls `\@beginlabelshook`.

```
\def\startlabels{%
  \clearpage%
  \pagestyle{empty}%
  \setlength{\topmargin}{-1.0in}%
  \if@envelope%
    \addtolength{\topmargin}{\EnvelopeTopMargin}%
    \else \addtolength{\topmargin}{\LabelTopMargin}%
```

```
\fi%
\setlength{\headheight}{0pt}%
\setlength{\headsep}{0pt}%
\setlength{\footskip}{0pt}%
\setlength{\textheight}{200in}%
\setlength\paperheight{\textheight}%
\global\vsize=200in\relax%
\addtolength{\textheight}{-\topmargin}%
\addtolength{\textheight}{-1.0in}%
\setlength{\oddsidemargin}{-1.0in}%
\if@envelope\relax%
\else%
   \addtolength{\oddsidemargin}{\LabelLeftMargin}%
\fi%
\setlength{\evensidemargin}{\oddsidemargin}%
\setlength{\textwidth}{20in}%
\hsize=20in%
\baselineskip=0pt%
\lineskip=0pt%
\parindent=0pt%
\if@envelope
```

Now we can calculate `\EnvelopeLeftMargin`

```
  \ifcase\the\@envelopeposition%
    \setlength{\EnvelopeLeftMargin}{\paperwidth}%
    \if@rotateenvelopes%
      \addtolength{\EnvelopeLeftMargin}{-\EnvelopeHeight}%
    \else%
      \addtolength{\EnvelopeLeftMargin}{-\EnvelopeWidth}%
    \fi%
    \setlength{\EnvelopeLeftMargin}{0.5\EnvelopeLeftMargin}%
  \or%
    \setlength{\EnvelopeLeftMargin}{0pt}%
  \or%
    \setlength{\EnvelopeLeftMargin}{\paperwidth}%
    \if@rotateenvelopes%
      \addtolength{\EnvelopeLeftMargin}{-\EnvelopeHeight}%
    \else%
      \addtolength{\EnvelopeLeftMargin}{-\EnvelopeWidth}%
    \fi%
  \else%
    \relax%
  \fi%
\else%
```

Initializing labels counters...

```
  \setcounter{LabelCountCol}{\theLabelOffsetCol}%
  \setcounter{LabelCountRow}{\theLabelOffsetRow}%
  \ifnum\theLabelOffsetRow>1%
    \null%
```

```
      \loop \vspace*{\LabelHeight}%
        \addtocounter{LabelOffsetRow}{-1} \ifnum\theLabelOffsetRow>1%
      \repeat%
    \fi%
    \ifnum\theLabelOffsetCol>1%
      \loop \hspace*{\LabelWidth}\nolinebreak%
        \addtocounter{LabelOffsetCol}{-1} \ifnum\theLabelOffsetCol>1%
      \repeat%
    \fi%
    \nopagebreak%
  \fi%
  \spaceskip0pt\relax%
  \xspaceskip 0pt\relax%
  \clubpenalty=0%
  \widowpenalty=0%
  \raggedbottom%
  \sloppy%
  \setlength\hfuzz{5in}%
  \setlength\vfuzz{5in}%
  \ignorespaces%
  \@beginlabelshook%
  \@beginlabelpagehook%
  \nopagebreak}%
```

7.6 Printing of the addresses

`\PrintReturnAddress` This macro uses the text as an argument and prints it according to the conventions.

```
\newcommand{\PrintReturnAddress}[1]{%
  \vspace*{\FromAddressTopMargin}
  \null\hspace{\FromAddressLeftMargin}
  \parbox[t][\FromAddressHeight]{\FromAddressWidth}%
    {\@fromaddressfont \lineskip=1pt
      \if@printreturnaddress #1\else\relax\fi}}
```

`\PrintAddress` This macro works like `\PrintReturnAddress`, with several important differences: it prints barcodes if necessary *and* capitalizes the address.

```
\newcommand{\PrintAddress}[1]{%
  \vspace*{\ToAddressTopMargin}
  \leavevmode
  \null\hspace*{\ToAddressLeftMargin}
  \parbox[t]{\ToAddressWidth}{%
    \lineskip=1pt
    \if@barcodes \PrintBarCode{#1} \fi
    \@toaddressfont
    \if@capitalizeaddress \@make@capitalize{#1} \else #1 \fi}}
```

7.7 Label setup

`\PrintLabel` This macro prints a label in a parbox

```
\newcommand{\PrintLabel}[1]{%
  \parbox[t][\LabelHeight]{\LabelWidth}{%
    \PrintAddress{#1}}}
```

`\PrintBigLabel` This macro makes a minipage with addresses on it, similarly to `\PrintEnvelope` below.

```
\newcommand{\PrintBigLabel}[2]{%
  \begin{minipage}[t][\LabelHeight]{\LabelWidth}%
    \baselineskip=0pt%
    \lineskip=0pt%
    \parindent=0pt%
    \begin{center}%
      \PrintReturnAddress{#1}\\%
      \rule{\ToAddressWidth}{0.1pt}%
      \PrintAddress{#2}%
    \end{center}%
  \end{minipage}}
```

7.8 Envelope setup

Labels include one box per label, so their setup is simple. The situation with envelopes is different: they contain several boxes, and could be rotated, centered, etc.

`\PrintEnvelope` This macro makes a minipage with addresses on it.

```
\newcommand{\PrintEnvelope}[2]{%
  \begin{minipage}[t][\EnvelopeHeight]{\EnvelopeWidth}%
    \baselineskip=0pt%
    \lineskip=0pt%
    \parindent=0pt%
    \PrintReturnAddress{#1}\\%
    \begin{center}%
      \PrintAddress{#2}%
    \end{center}%
  \end{minipage}}
```

`\@PrintEnvelope` The following macro checks for rotation:

```
\newcommand{\@PrintEnvelope}[2]{%
  \if@rotateenvelopes\rotatebox{90}{\PrintEnvelope{#1}{#2}}%
  \else\PrintEnvelope{#1}{#2}%
  \fi}
```

8 Printing of envelopes and labels

8.1 Main Command

`\mlabel` Now we are prepared to print actual envelopes and labels. It is done by the `\mlabel` command. It has two forms: for labels and envelopes.

```
\renewcommand{\mlabel}[2]{\ignorespaces%
  \spaceskip 0pt\relax%
  \xspaceskip 0pt\relax%
```

8.2 Printing of one envelope

```
  \if@envelope%
    \leavevmode%
    \hspace*{\EnvelopeLeftMargin}%
    \@PrintEnvelope{#1}{#2}%
    \clearpage%
    \@beginlabelpagehook%
```

8.3 Printing of one label

```
  \else%
    \ignorespaces%
    \ifnum\theLabelCountCol>\theLabelMaxCol%
      \\\nopagebreak%
      \stepcounter{LabelCountRow}%
      \setcounter{LabelCountCol}{1}%
    \fi%
    \ifnum\theLabelCountRow>\theLabelMaxRow%
      \vfill\eject\@beginlabelpagehook%
      \setcounter{LabelCountRow}{1}%
      \setcounter{LabelCountCol}{1}%
    \fi%
    \if@biglabel%
        \PrintBigLabel{#1}{#2}%
    \else%
        \PrintLabel{#2}%
    \fi%
    \ignorespaces\nolinebreak%
    \stepcounter{LabelCountCol}%
  \fi}%
```

8.4 Printing of return labels

We print only mailing addresses on labels. The user is supposed to have preprinted return labels. Here we describe a utility for printing them. This utility should be used in a separate document.

`\@numreturnlabels` The counter `\@numreturnlabels` stores the number of return labels to be printed. Note that it is a TEX counter, not a LATEX one.

```
\newcount\@numreturnlabels
```

`\printreturnlabels` This macro has two parameters: the number of labels to be printed and the text that is printed. It is the same on all labels.

```
\newcommand{\printreturnlabels}[2]{%
  \@numreturnlabels=#1
  \def\@toaddressfont{\@fromaddressfont}
```

```
\@capitalizeaddressfalse
\@barcodesfalse
\startlabels
\loop \mlabel{\relax}{#2} \advance\@numreturnlabels by -1
  \ifnum\@numreturnlabels>0\repeat}
```

9 Barcodes

9.1 Main command

The USPS Postnet codes are printed accordingly to the specifications Ref. [3]. The scanning algorithm is stolen from David Carlisle's enumerate package [1].

\PrintBarCode First, we extract barcodes by the command `\@extractbarcode`. Then we print them by `\@printbarcode`.

```
\newcommand{\PrintBarCode}[1]{%
  \@extractbarcode{#1}
  \@printbarcode}
```

9.2 Extraction of barcodes

We define zipcode as a sequence of digits (0–9) that:

- Has no characters other than digits and dashes (-) inside it
- Has no bracketed groups inside it and is not bracketed itself
- Is the last in the address field unless `\if@alwaysbarcodes=true`

We print this sequence plus the *control character*. The latter is defined as minus sum of digits of the zip code modulo 10 (that is, the complement of the sum of digits to a multiple of 10).

\@zipcode \@zipcodesum \@zipcodefound First, some internal registers. The token list `\@zipcode` contains barcode found so far. The register `\@zipcodesum` first contains the sum of digits of the barcode, and then the control character. The switch `\@zipcodefound` shows whether we found zip code so far.

```
\newtoks\@zipcode
\newcount\@zipcodesum
\newif\if@zipcodefound
```

There are two modes for gobbling tokens:

State A: We are outside a potential zipcode sequence (`\if@zipcodefound=false`)

State B: We are inside a potential zipcode sequence (`\if@zipcodefound=true`)

\@endaddress \@finishzipcode
- If we meet in any state the special token `\@endaddress`, we gobble it and finish the loop.

```
  \long\def\@finishzipcode#1{}
```

\@firstzipcode

- If we meet a number (0–9) in state A, we initialize registers, process the token and go to state B

```
\long\def\@firstzipcode#1{%
  \@zipcode{#1}
  \@zipcodesum=#1\relax
  \@zipcodefoundtrue
  \@zipcodeloop}
```

\@continuezipcode

- If we meet a number (0–9) in state B, we just process it.

```
\long\def\@continuezipcode#1{%
  \@zipcode=\expandafter{\the\@zipcode#1}
  \advance\@zipcodesum by #1
  \@zipcodeloop}
```

\@dashzipcode

- If we meet a dash in state B, we gobble it.

```
\long\def\@dashzipcode#1{\@zipcodeloop}
```

\@spacezipcode

- If we meet a space in any state, we gobble it and go to the state A. The trick is from Carlisle's enumerate package.

```
\def\@spacezipcode{%
  \@zipcodefoundfalse
  \afterassignment\@zipcodeloop\let\EL@temp= }
```

\@abortzipcode

- If we meet anything else in any mode, we gobble it and go to state A

```
\long\def\@abortzipcode#1{%
  \@zipcodefoundfalse
  \@zipcodeloop}
```

\@zipcodeloop
\EL@temp

This macro is simple. We just put the next token into \EL@temp and process it through \@zipcodeloop@.

```
\def\@zipcodeloop{\futurelet\EL@temp\@zipcodeloop@}
```

\@zipcodeloop@
\EL@tempa

This macro performs actual processing... We put the command that gobbles the next token into \EL@tempa

```
\def\@zipcodeloop@{%
 \ifx \@endaddress\EL@temp      \def\EL@tempa{\@finishzipcode}    \else
 \ifx 0\EL@temp \if@zipcodefound \def\EL@tempa{\@continuezipcode}
              \else           \def\EL@tempa{\@firstzipcode} \fi \else
 \ifx 1\EL@temp \if@zipcodefound \def\EL@tempa{\@continuezipcode}
              \else           \def\EL@tempa{\@firstzipcode} \fi \else
 \ifx 2\EL@temp \if@zipcodefound \def\EL@tempa{\@continuezipcode}
              \else           \def\EL@tempa{\@firstzipcode} \fi \else
 \ifx 3\EL@temp \if@zipcodefound \def\EL@tempa{\@continuezipcode}
              \else           \def\EL@tempa{\@firstzipcode} \fi \else
 \ifx 4\EL@temp \if@zipcodefound \def\EL@tempa{\@continuezipcode}
              \else           \def\EL@tempa{\@firstzipcode} \fi \else
 \ifx 5\EL@temp \if@zipcodefound \def\EL@tempa{\@continuezipcode}
```

```
              \else           \def\EL@tempa{\@firstzipcode} \fi \else
  \ifx 6\EL@temp \if@zipcodefound \def\EL@tempa{\@continuezipcode}
              \else           \def\EL@tempa{\@firstzipcode} \fi \else
  \ifx 7\EL@temp \if@zipcodefound \def\EL@tempa{\@continuezipcode}
              \else           \def\EL@tempa{\@firstzipcode} \fi \else
  \ifx 8\EL@temp \if@zipcodefound \def\EL@tempa{\@continuezipcode}
              \else           \def\EL@tempa{\@firstzipcode} \fi \else
  \ifx 9\EL@temp \if@zipcodefound \def\EL@tempa{\@continuezipcode}
              \else           \def\EL@tempa{\@firstzipcode} \fi \else
  \ifx -\EL@temp \if@zipcodefound \def\EL@tempa{\@dashzipcode}
              \else           \def\EL@tempa{\@abortzipcode} \fi \else
  \ifx \@sptoken\EL@temp         \def\EL@tempa{\@spacezipcode}     \else
                              \def\EL@tempa{\@abortzipcode}
  \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
  \EL@tempa}
```

`\@extractbarcode` The command `\@extractbarcode` puts barcode into the `\@zipcode`, and calculates the control character (10 minus sum of the digits of the barcode).

```
\long\def\@extractbarcode#1{%
  \@zipcodefoundfalse
  \@zipcodeloop#1\@endaddress
  \if@alwaysbarcodes \@zipcodefoundtrue \fi
  \if@zipcodefound
    \ifnum\the\@zipcodesum>0
      \loop \advance \@zipcodesum by -10 \ifnum\the\@zipcodesum>0
      \repeat
    \fi
    \multiply\@zipcodesum by -1
  \fi}
```

9.3 Printing barcodes

`\@barcodewidth` `\@barcodeLheight` `\@barcodeSheight` `\@barcodeskip` First, some lengths. "L" and "S" below refer to "long" and "short" bars correspondingly.

```
\newlength{\@barcodewidth}
\newlength{\@barcodeLheight}
\newlength{\@barcodeSheight}
\newlength{\@barcodeskip}
\setlength{\@barcodewidth}{0.020in}
\setlength{\@barcodeLheight}{0.125in}
\setlength{\@barcodeSheight}{0.050in}
\setlength{\@barcodeskip}{0.026in}
```

`\@barL` `\@barS` The following macros print long and short bars.

```
\DeclareRobustCommand{\@barL}{%
  \rule{\@barcodewidth}{\@barcodeLheight}\hspace{\@barcodeskip}}
\DeclareRobustCommand{\@barS}{%
  \rule{\@barcodewidth}{\@barcodeSheight}\hspace{\@barcodeskip}}
```

`\@printonezip` `\@printbarcode` The scanning of `\@zipcode` is simpler than the scanning of the address: the

only tokens we can meet are digits. Well, we will add an end marking token to the list. Let it be the letter "S" (from "Stop").

```
\def\@printonezip#1{%
  \ifx1#1\@barS\@barS\@barS\@barL\@barL\else
  \ifx2#1\@barS\@barS\@barL\@barS\@barL\else
  \ifx3#1\@barS\@barS\@barL\@barL\@barS\else
  \ifx4#1\@barS\@barL\@barS\@barS\@barL\else
  \ifx5#1\@barS\@barL\@barS\@barL\@barS\else
  \ifx6#1\@barS\@barL\@barL\@barS\@barS\else
  \ifx7#1\@barL\@barS\@barS\@barS\@barL\else
  \ifx8#1\@barL\@barS\@barS\@barL\@barS\else
  \ifx9#1\@barL\@barS\@barL\@barS\@barS\else
  \ifx0#1\@barL\@barL\@barS\@barS\@barS\else
  \ifx S#1\def\EL@tempa{\relax}%
  \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi%
  \EL@tempa}
\def\@printbarcode{%
  \if@zipcodefound
   \mbox{%
    \@barL%
    \def\EL@tempa{\@printonezip}%
    \expandafter\EL@tempa\the\@zipcode S%
    \def\EL@tempa{\@printonezip}%
    \expandafter\EL@tempa\the\@zipcodesum S%
    \@barL}
    \\[1ex]
  \fi}
```

10 Capitalization

These macros process the address (actually, any string) according to the USPS recommendations. Specifically, they:

- Strip dots (.) and commas (,) from the address unless they are enclosed in brackets
- Make all letters uppercase
- Add 1pt space between letters
- Add 1em space between words

An interesting question is whether we should de-accent accented letters. USPS says nothing about it. In the present version accents are *not* stripped. However due to the scanning algorithm they should be enclosed by brackets, like this: `{\u S}andor {\c C}edi`.

`\@addr@cap` We store the capitalized address in the token list `\@addr@cap`.

```
\newtoks\@addr@cap
```

The following macros process the tokens one by one.

\@finishaddrcap If we meet the special token \@endaddress, we gobble it and stop.

```
\long\def\@finishaddrcap#1{}
```

\@dotcommaaddrcap If we meet comma or dot, we gobble it and do *not* stop. This macro is also useful for gobbling LaTeX 2ε letter commands like \voidb@x and \unhbox.

```
\long\def\@dotcommaaddrcap#1{%
  \@addrcaploop}
```

\@newlineaddrcap If we meet \\, we add it to the list

```
\long\def\@newlineaddrcap#1{%
  \@addr@cap=\expandafter{\the\@addr@cap #1}
  \@addrcaploop}
```

\@bgroupaddrcap If we meet \bgroup, we add it to the list the complete group (uppercase)

```
\long\def\@bgroupaddrcap#1{%
  \@addr@cap=\expandafter{\the\@addr@cap {\MakeUppercase{#1}}}
  \@addrcaploop}
```

\@spaceaddrcap If we meet a space we gobble it (oh-oh) and add it to the list.

```
\def\@spaceaddrcap{%
  \@addr@cap=\expandafter{\the\@addr@cap\hspace{0.6em}}
  \afterassignment\@addrcaploop\let\EL@temp= }
```

\@otheraddrcap And if we meet anything else, we make it uppercase and add to the list

```
\def\@otheraddrcap#1{%
  \@addr@cap=\expandafter{\the\@addr@cap%
     \MakeUppercase{#1}\kern1pt\relax}
  \@addrcaploop}
```

\@addrcaploop This macro is simple. We just put the next token into \EL@temp and process it through \@addrcaploop@.

```
\def\@addrcaploop{\futurelet\EL@temp\@addrcaploop@}
```

\@addrcaploop@ This macro performs actual processing...

```
\def\@addrcaploop@{%
  \ifx \@endaddress\EL@temp      \def\EL@tempa{\@finishaddrcap}    \else
  \ifx .\EL@temp                 \def\EL@tempa{\@dotcommaaddrcap}  \else
  \ifx ,\EL@temp                 \def\EL@tempa{\@dotcommaaddrcap}  \else
  \ifx \voidb@x\EL@temp          \def\EL@tempa{\@dotcommaaddrcap}  \else
  \ifx \unhbox\EL@temp            \def\EL@tempa{\@dotcommaaddrcap} \else
  \ifx \\\EL@temp                \def\EL@tempa{\@newlineaddrcap}   \else
  \ifx \bgroup\EL@temp           \def\EL@tempa{\@bgroupaddrcap}    \else
  \ifx \@sptoken\EL@temp         \def\EL@tempa{\@spaceaddrcap}     \else
                                \def\EL@tempa{\@otheraddrcap}
  \fi\fi\fi\fi\fi\fi\fi\fi
  \EL@tempa}
```

\@make@capitalize

```
\long\def\@make@capitalize#1{%
  \@addr@cap={\relax}
  \@addrcaploop#1\@endaddress
  \the\@addr@cap}
```

11 Games with .aux file

The commands described in this section write something to the `.aux` file, and thus change the way *EnvLab* treats the labels and envelopes. Since the action is delayed till the `.aux` file is processed, these commands affect only the labels automatically extracted from the `letter` environment, and do not affect the labels explicitly defined by the `\mlabel` commands in the main file.

`\@@mlabel` The macro `\@@mlabel` stores the status of the `\@mlabel` as determined by `\makelabels`. It is a no-op at the start. If `\makelabels` redefines `\@mlabel`, we catch it through the `\AtEndDocument` hook. Note that since `\makelabel` is allowed only in the preamble, we are not in danger of redefining commands too early.

```
\let\@@mlabel=\@gobbletwo
\AtEndDocument{\let\@@mlabel=\@mlabel}
```

The next four commands redefine `\@mlabel` to suppress or resume printing mailing labels.

`\suppresslabels` This command suppress printing labels and envelopes until it is resumed by `\resumelabels` or similar commands.

```
\def\suppresslabels{\if@filesw\immediate\write\@auxout{%
  \string\@suppresslabels}\fi}
```

`\@suppresslabels` This is the internal command that performs the actual processing.

```
\def\@suppresslabels{\let\@mlabel=\@gobbletwo}
```

`\resumelabels` `\@resumelabels` These commands resume printing labels and envelopes if it was suppressed by `\suppresslabels` or similar commands.

```
\def\resumelabels{\if@filesw\immediate\write\@auxout{%
  \string\@resumelabels}\fi}
\def\@resumelabels{\let\@mlabel=\@@mlabel}
```

`\suppressonelabel` `\@suppressonelabel` `\@old@mlabel` These commands suppress printing of one label or envelope. The macro `\@old@mlabel` is used to store the system state.

```
\def\suppressonelabel{\if@filesw\immediate\write\@auxout{%
  \string\@suppressonelabel}\fi}
\def\@suppressonelabel{\let\@old@mlabel=\@mlabel%
  \def\@mlabel{%
    \let\@mlabel=\@old@mlabel%
    \@gobbletwo}}
```

`\printonelabel` `\@printonelabel` These commands resume printing for one label or envelope The macro `\@old@mlabel` is used to store the system state.

```
\def\printonelabel{\if@filesw\immediate\write\@auxout{%
  \string\@printonelabel}\fi}
\def\@printonelabel{\let\@old@mlabel=\@mlabel%
  \def\@mlabel{%
    \let\@mlabel=\@old@mlabel%
    \@@mlabel}}
```

\ChangeEnvelope
\@ChangeEnvelope
\@ChangeEnvelopeStar

This macro writes `\@SetEnvelope` to the `.aux` file. It has two forms: starred and and unstarred. In the unstarred mode it also writes `\@startlabels` to the `.aux` file. In the unstarred form it does not. Since we want to treat *both* stars and optional arguments, we introduce two internal commands that can be invoked by the main macro.

```
\def\ChangeEnvelope{\@ifstar{\@ChangeEnvelopeStar}{\@ChangeEnvelope}}
\newcommand\@ChangeEnvelopeStar[3][0pt]{%
  \if@filesw\immediate\write\@auxout{%
       \string\@SetEnvelope[#1]{#2}{#3}}%
   \fi}
\newcommand\@ChangeEnvelope[3][0pt]{%
  \if@filesw\immediate\write\@auxout{%
       \string\@SetEnvelope[#1]{#2}{#3}}
     \immediate\write\@auxout{\string\@startlabels}
  \fi}
```

\@SetEnvelope

We define this command as no-op at beginning, and then redefine it before reading `.aux` file.

```
\def\@SetEnvelope[#1]#2#3{}
\AtEndDocument{\let\@SetEnvelope=\SetEnvelope}
```

\ChangeLabel
\@ChangeLabel
\@ChangeLabelStar

This macro writes `\@SetLabel` to the `.aux` file. It has two forms: starred and and unstarred. In the unstarred mode it also writes `\@startlabels` to the `.aux` file. In the unstarred form it does not. Since we want to treat *both* stars and optional arguments, we introduce two internal commands that can be invoked by the main macro.

```
\def\ChangeLabel{\@ifstar{\@ChangeLabelStar}{\@ChangeLabel}}
\newcommand\@ChangeLabelStar[7]{%
  \if@filesw\immediate\write\@auxout{%
       \string\@SetLabel{#1}{#2}{#3}{#4}{#5}{#6}{#7}}%
   \fi}
\newcommand\@ChangeLabel[7]{%
  \if@filesw\immediate\write\@auxout{%
       \string\@SetLabel{#1}{#2}{#3}{#4}{#5}{#6}{#7}}
     \immediate\write\@auxout{\string\@startlabels}
  \fi}
```

\@SetLabel

We define this command as no-op at beginning, and then redefine it before reading `.aux` file.

```
\def\@SetLabel#1#2#3#4#5#6#7{}
\AtEndDocument{\let\@SetLabel=\SetLabel}
```

\ChangeBigLabel
\@ChangeBigLabel
\@ChangeBigLabelStar

This macro writes `\@SetBigLabel` to the `.aux` file. It has two forms: starred and and unstarred. In the unstarred mode it also writes `\@startlabels` to the `.aux` file. In the unstarred form it does not. Since we want to treat *both* stars and optional arguments, we introduce two internal commands that can be invoked by the main macro.

```
\def\ChangeBigLabel{\@ifstar{\@ChangeBigLabelStar}{\@ChangeBigLabel}}
```

```
\newcommand\@ChangeBigLabelStar[7]{%
  \if@filesw\immediate\write\@auxout{%
       \string\@SetBigLabel{#1}{#2}{#3}{#4}{#5}{#6}{#7}}%
   \fi}
\newcommand\@ChangeBigLabel[7]{%
  \if@filesw\immediate\write\@auxout{%
       \string\@SetBigLabel{#1}{#2}{#3}{#4}{#5}{#6}{#7}}
     \immediate\write\@auxout{\string\@startlabels}
  \fi}
```

`\@SetLabel` We define this command as no-op at beginning, and then redefine it before reading `.aux` file.

```
\def\@SetBigLabel#1#2#3#4#5#6#7{}
\AtEndDocument{\let\@SetBigLabel=\SetBigLabel}
```

12 Reimplementation of the `\opening` command

`\re` Some people like to put below the address information like

> Re: our recent talk

A way to do this is to include it in the address like this:

```
\begin{letter}{%
  Dr.~Austin Tankel\\
  Some University\\
  Anytown, Pa 12345\\[1ex]
  Re: Our recent talk}
\opening{Dear Austin:}
```

However, this additional info will be put in the mailing label, which is wrong. Here we describe a macro that works like this:

```
\begin{letter}{%
  Dr.~Austin Tankel\\
  Some University\\
  Anytown, Pa 12345}
\re{Our recent talk}
\opening{Dear Austin:}
```

Now, the implementation. First, lets us check whether the option `re` is chosen (otherwise we don't bother to redefine the commands):

```
\if@EL@redefine@opening
```

`\re` `\recontentents` The command `\re` just defines `\recontents`. Also, we initialize `\recontents` to be initially empty

```
  \newcommand*{\re}[1]{\def\recontents{#1}}%
```

\ReName By default it is just plain style “Re: ” (note the space!)

```
\def\ReName{Re: }%
```

\opening Now we redefine the standard \opening command to include the \re info. Here is the quote from the standard letter class [2]:

> Text is begun with the \opening command, whose argument generates the salutation, as in
>
> ```
> \opening{Dear Henry,}
> ```
>
> This should produce everything up to and including the ‘Dear Henry,’ and a \par command that follows. Since there’s a \vfil at the bottom of every page, it can add vertical fill to position a short letter. It should use the following commands:
>
> - \toname : name part of ‘to’ address. Will be one line long.
> - \toaddress : address part of ‘to’ address. The lines separated by \\.
> - \fromname : name of sender.
> - \fromaddress : argument of current \address declaration– null if none. Should use standard institutional address if null.
> - \fromlocation : argument of current \location declaration– null if none.
> - \telephonenum : argument of current \telephone declaration– null if none.

We just \ReName and \recontents here...

```
\renewcommand*{\opening}[1]{\ifx\@empty\fromaddress
  \thispagestyle{firstpage}%
    {\raggedleft\@date\par}%
  \else  % home address
    \thispagestyle{empty}%
    {\raggedleft\begin{tabular}{l}\ignorespaces
        \fromaddress \\*[2\parskip]%
        \@date \end{tabular}\par}%
  \fi
  \vspace{2\parskip}%
  {\raggedright \toname \\ \toaddress \par}%
  \ifx\@empty\recontents\relax
  \else
     {\raggedright \ReName \recontents \par}%
  \fi
  \vspace{2\parskip}%
  #1\par\nobreak}%
```

Now we close \if@EL@redefine@opening:

```
\fi
```

And the last line:

```
⟨/package⟩
```

References

[1] David Carlisle. *The enumerate package*. CTAN, v2.02 edition, January 1994.

[2] Leslie Lamport, Frank Mittelbach, and Rainer Schöpf. Standard letter document class for LaTeX version 2e. CTAN, 199c.

[3] USPS. *Designing Business Letter Mail (Pub 25)*, August 1995.

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

R

S

T

V

W