A package for making sticky labels in LaTeX*

Sebastian Rahtz, Leonor Barroca   Grant Gustafson

1995/06/22

Contents

Abstract

A LaTeX style to print a regular grid of ragged-right labels on a page, suitable for sheets of labels which can be fed through a laser printer. Macros are provided to allow easy input of names and addresses in a form free of TeX markup. Equally useful is a feature for making multiple copies of a single label, e.g., return address stickers to go with the labels.

1 Usage

This style file was written to produce a sheet of labels which can be xeroxed onto Avery brand 5360 sticky-label material. This label material is made for a xerox machine and it has 7 rows and 3 columns of labels on `letterpaper`. It works for other label layouts as well. In particular, the defaults for the most popular paper sizes `letterpaper` and `a4paper` are 7 rows by 3 columns and 8 rows by 3 columns, respectively.

Other uses include production of disk labels, book labels, shipping labels, name tags, door cards, photo galleries and compact membership lists.

The service of this *package* is to produce a rectangular grid of items on a sheet of paper, with each item centered in its grid area. It really doesn't matter what

*This file has version number 10, last revised 1995/06/22.

goes in the grid area: text, graphics or any LaTeX construct. For example, you might use the `fancybox` package to produce oval-box or shadow-box name tags for a conference.

1.1 Paper Sizes

The paper size is supplied by the document options for the class. The European default is `a4paper` and the American default is `letterpaper`, 8.5×11-inch, inherited from the class file, e.g., `article.cls`.

a4paper 297mm by 210mm.

a5paper 210mm by 148mm.

b5paper 250mm by 176mm.

letterpaper 11in by 8.5in.

legalpaper 14in by 8.5in.

executivepaper 10.5in by 7.25in.

1.2 Label Stock Sizes

A number of different label stocks are available for xerox machines and laser printers. The typical stock has m rows and n columns with various gutters on the page. Below is a table of Avery products that are in common use in America.

Label Size in inches	Labels per page	Rows	Cols	Gutters	Avery Stock No
$1\frac{7}{16} \times 2\frac{13}{16}$	21/page	7	3	Yes	5360
$2 \times 4\frac{1}{4}$	10/page	5	2	Yes	5352
2×4	10/page	5	2	Yes	5327
1×4	20/page	10	2	Yes	5161, 5261
$1\frac{3}{8} \times 2\frac{11}{16}$	21/page	7	3	Yes	5329
$1 \times 2\frac{5}{8}$	30/page	10	3	Yes	5331, 5160, 5260
$1\frac{1}{3} \times 4$	14/page	7	2	Yes	5162, 5262
$\frac{1}{2} \times 1\frac{3}{4}$	80/page	20	4	Yes	5267
$1 \times 2\frac{3}{4}$	33/page	11	3	No	5332, 5351, 5354, 5314

Another common label stock is Dennison 37-141, having 3 columns and 8 rows of $1\frac{3}{8} \times 2\frac{7}{8}$ gutterless labels on letterpaper. The information by the manufacturer is not precisely correct, because the first column *is* exactly $2\frac{7}{8}$ inches wide, but the other two are $2\frac{13}{16}$ inches wide.

A common problem with implementing this package on a new label stock is precision of the laser printer, which prints the master, and the xerox machine which prints the master onto the label stock. Some of the above are for direct use on a laser printer, which removes one source of error, but replaces it by the possibility that the label stock will jam in the laser printer!

1.3 Customizing for Other Labels

Your labels will very likely be different than Avery 5360. So first you have to tailor your source file to a particular type of label. The lines below may be placed in the preamble and edited to suit the application and the actual printer used for output.

```
\documentclass[12pt]{article}
\usepackage{labels}
\LabelCols=3%       Number of columns of labels per page
\LabelRows=7%       Number of rows of labels per page
\LeftBorder=8mm%    Space added to left border of each label
\RightBorder=8mm%   Space added to right border of each label
\TopBorder=9mm%     Space to leave at top of sheet
\BottomBorder=2mm%  Space to leave at bottom of sheet
\begin{document}%   End of preamble
```

These controls can also be issued after the beginning of the document, but the results are undefined unless the following example is followed, which uses the `\LabelSetup` control sequence:

```
\begin{document}%   End of preamble
\LabelCols=3%       Number of columns of labels per page
\LabelRows=7%       Number of rows of labels per page
\LeftBorder=8mm%    Space added to left border of each label
\RightBorder=8mm%   Space added to right border of each label
\TopBorder=9mm%     Space to leave at top of sheet
\BottomBorder=2mm%  Space to leave at bottom of sheet
\LabelSetup%        Invoke new settings
```

For example, *your* grid maybe has only two columns of ten labels each, requiring the first two variables to be reset. The gutter areas on the label material dictate how to set the other parameters, plus of course personal adjustments of the amount of white space around each label.

Your printer driver is expected to print the page *exactly* as it should in vanilla TEX, i.e., with the origin of the page down one inch and right one inch from the top left hand corner of the paper. If it doesn't, adjust your driver parameters.

The usual problem with printers is that label output is quite wide, requiring the very edges of the paper, on which your printer may not write correctly. Adjust the parameters `\LeftBorder`, `\RightBorder`, `\TopBorder`, `\BottomBorder` as outlined below to solve this common problem.

First of all, `\TopBorder` is not the white space at the top of the page, but the amount of space to leave at the top so that the first label is centered on its grid. This number is determined by trial and error using the actual printer, xerox machine and label stock to be used in the production run.

The second value `\BottomBorder` is determine empirically also, so that the labels are all centered on their grids. A mathematical formula to be satisfied is

$$\texttt{Label Height} = \frac{\texttt{paperheight} - \texttt{TopBorder} - \texttt{BottomBorder}}{\texttt{LabelRows}}$$

The values `\LeftBorder` and `\RightBorder` are amounts of white space to be added to the left and right of the actual label area so that the label itself does not smash into the edge of the grid. The actual label width is considerably smaller than the grid width (physical width of the label stock). It is determined by the formula

$$\texttt{LeftBorder} + \texttt{Label Width} + \texttt{RightBorder} = \frac{\texttt{paperwidth}}{\texttt{LabelCols}}$$

Normally, the left and right white space dimensions are the same, but there can be exceptions. It is best to determine the dimensions empirically on the actual equipment to be used in the production run. Visual guides from a `dvi` viewer can help, but be aware that the laser printer may fail to print near the edge and the xerox machine may either enlarge or shrink the image onto the label stock.

An option `grid overlay` can be produced with the label output for testing purposes by the preamble control sequence `\LabelInfotrue`. Most `dvi` viewers are capable of showing the entire test grid. Once printed, the overlay can be compared with the actual label stock to see if the borders are in the proper place. By adjusting parameters, the master copy off the laser printer should be able to go through the xerox machine automatically.

1.4 Using an External Label source File

The simplest form of input is very easy, as in the following example:

```
\documentclass{article}
\usepackage{labels}
\begin{document}
\labelfile{names.dat}
\end{document}
```

where `names.dat` contains names and address in *plain format*. Source files in plain format may contain extra blank lines (only one blank between labels is required). Sort fields should begin with `%` in column 1, so they don't print as part of the label. Leading and trailing blanks on lines are ignored. Ctrl-L's seem to be acceptable in the source. They are not ignored, even if one appears at the end of the file. For example, if `^L` appears on a line by itself, then it either produces an empty label or else a blank line, if it happens to be in a group of label lines. The label data commonly appears on lines, flush left, with no intervening blank lines, because a blank line signals a new label. To purposely create blank labels, as for filling out a page of labels, see the control sequence `\skiplabels{#1}`, *infra*. The actual text in the external file may contain LaTeX markup controls. This includes comment marks `%`, which will be ignored during typesetting.

Source File Creation and Sorting

Most mailing lists that already exist can be edited to create an acceptable list in the proper format. If you want long lines to wrap, then force all the material to be wrapped onto one line in the source. Otherwise, break the material into lines of the length desired. Any special items in a mailing label can be coded in LaTeX, e.g., foreign names with accented characters, trademarks and font size changes.

Generally, raw sources have to be searched for special LaTeX controls and brought up to standard. The following characters are trouble in a raw source file. Only use for writing LaTeX code obeying LaTeX coding rules!

Double quote	\"	Hash (number)	\#	Underscore	_
Dollar	\$	Percent	\%	Ampersand	\&
Less than	\<	Greater than	\>	Vertical bar	\\|
Commercial at	\@	Backslash	\\	Circumflex	\^
Left brace	\{	Right brace	\}	Tilde	\~

A typical comment line might start with `%` and then `^L` (ctrl-L) followed by a sort key (e.g., the zip code or last name). The `emacs` editor supports a method of sorting such records, thereby rearranging the source file into a new label order. The method:

- Mark the whole file as a region: `^[<` goes to the top of file, then `^@` sets mark; follow by `^[>` to mark the whole file.
- Invoke `emacs`'s `sort-pages` external routine as follows. Press `^[x` then enter `sort-pages` and press return. The region marked will be sorted on the first line of each page, with a page delimited by `^L`.

General purpose LaTeX Source

If you want to set up a general-purpose LaTeX source file, then use the following source, which prompts for the labels file name.

```
\documentclass{article}
\usepackage{labels}
\begin{document}
\promptlabels
\end{document}
```

1.5 Labels in the Main File

You can, of course, just have the names and addresses in the main file, rather than using `\labelfile` to include them. The file format is to be exactly as described above, especially, follow the advice about blank lines, which delimit labels. This example makes use of the `labels` environment.

```
\documentclass{article}
\usepackage{labels}
\begin{document}
\begin{labels}
Me
My address
My City, State, Zipcode

My Brother
His address
His City, State, Zipcode
\end{labels}
\end{document}
```

1.6 Control Sequences to Make Labels

There are other ways of accessing the same system.

1. The control sequence called `\addresslabel[#1]{#2}` accepts for optional argument `#1` LaTeX controls, e.g., font size and style, and for argument `#2` the rows of a `tabular` array of one column.

   ```
   \documentclass{article}
   \usepackage{labels}
   \begin{document}
     \addresslabel[\small\sf]
     {Me \\my street \\ mytown \\ England}
   \end{document}
   ```

2. Boxed labels use the special macro called `\boxedaddresslabel[#1]{#2}`, as follows. This produces a frame-box around the label itself, leaving lots of white space around the frame. An optional argument `[#1]` is provided for local setting of LaTeX controls, for example, `[\fboxsep=3pt]` will change the white space near the frame. The boolean variable `\LabelInfotrue` appears in the preamble in order to print out additional information, especially the label dimensions and the settings of certain variables.

   ```
   \documentclass{article}
   \usepackage{labels}
   \LabelInfotrue
   \begin{document}
     \boxedaddresslabel[\fboxsep=3pt]
     {{\bf Me} \\my street \\ mytown \\ England}
   \end{document}
   ```

3. To *duplicate* a label, there is a counter called `\numberoflabels` which you can set. For example, to print a return address 21 times, use this source:

   ```
   \documentclass{article}
   \usepackage{labels}
   \numberoflabels=21
   \begin{document}
     \addresslabel{Me \\my street \\ mytown \\ England}
   \end{document}
   ```

4. For more sophisticated users, there is a macro `\genericlabel` which you can call, with an argument of whatever you want to appear on the label (e.g., for disk labels). Thus you could have

```
\genericlabel{%
\begin{tabular}{|c|}
\hline
My Amazing Program\\
\hline
Disk 1 of 1\\
\hline
\em We aim to serve\\
 \hline
\end{tabular}
}
```

to produce a label like this:

```
|------------------|
|My Amazing Program|
|------------------|
|Disk 1 of 1       |
|------------------|
|We aim to serve   |
|------------------|
```

This feature has been used to print business cards with graphical logo.

1.7 Wrapping Long Lines and Debugging

Debugging of label files can be assisted by the internal error messages which are emitted when a label box is too high or too wide for the set parameters. This kind of error is unlikely to occur with the standard controls and the `labels` environment, because by default they use ragged right and wrap long lines.

In all modes, you can opt for a grid around each label field by setting a Boolean variable called `\LabelGridtrue`, e.g.,

```
\documentclass{article}
\usepackage{labels}
\LabelGridtrue
\numberoflabels=21
\begin{document}
  \addresslabel{Me \\my street \\ mytown \\ England}
\end{document}
```

By default you get no grids. The grids are useful for judging the 'spillover' of addresses onto adjacent labels, caused by long lines. See also `\boxedaddresslabel`, which draws a tighter box with more white space around the label text. Both can be used at the same time.

2 The macros

First of all, identify the package start, and provide some compatibility code for LaTeX 2.09. The general agreement is that macro packages will not be retrofitted to the now obsolete LaTeX 2.09, but this one happens to work.

```
⟨*package⟩
%
⟨!latex209⟩\ProvidesPackage{labels}[\LabelsFileInfo]
⟨!latex209⟩\NeedsTeXFormat{LaTeX2e}
⟨*latex209⟩
\newdimen\paperwidth
\paperwidth=8.5in% 297mm for a4paper
\newdimen\paperheight
\paperheight=11in% 210mm for a4paper
\def\settoheight#1#2{\setbox\@tempboxa%
```

```
  \hbox{#2}#1\ht\@tempboxa\setbox\@tempboxa\box\voidb@x}
\def\PackageWarning#1#2{\typeout{#1: #2}}
⟨/latex209⟩
```

We will be recording the size of a label, and the dimensions of the grid, so set up variables accordingly.

```
\newcount\LabelCols
\newcount\LabelRows
\newdimen\LeftBorder
\newdimen\RightBorder
\newdimen\TopBorder
\newdimen\BottomBorder
\newcount\numberoflabels
\newdimen\label@width
\newdimen\label@height
\newdimen\half@label
\newdimen\area@width
\newdimen\addr@width
\newdimen\LabTmp
\newsavebox{\this@label}
\newcount\label@number
\newcount\skip@labels
\newcount\l@so@far
\newif\ifLabelGrid
\newif\iffirst@label
\newif\ifLabelInfo
\first@labeltrue
\LabelGridfalse
\LabelInfofalse
```

Set defaults for the labels based upon paper size and common use. These values can be reset dynamically at runtime in the preamble.

```
\ifdim\paperwidth=210mm\relax%
   \LabelCols=3\relax\LabelRows=8\relax% a4paper
\else
   \LabelCols=3\relax\LabelRows=7\relax% letterpaper
\fi%
```

Variables are provided to allow you to force a border on the left and right edges of each label. The values will affect every label, of course, so you may need to experiment to get pleasing results. The other variables adjust the gutter width at the top and bottom of a page. They apply just to these two edges, and do not apply to a particular label. A LaserJetIII seems to ignore about 8mm on the edges. Xerox machines ignore even more, on all sides.

```
\LeftBorder=8mm
\RightBorder=8mm
\TopBorder=9mm
\BottomBorder=2mm
```

We need to reset all the dimensions appropriately for a page of labels, and the printer will need to know about the paper size as well.

```
\textwidth=\paperwidth
\textheight=\paperheight
\topmargin=-1in
\headheight=0em
```

```
\headsep=0em
\topskip=0em
\footskip=0em
\oddsidemargin=-1in
\evensidemargin=-1in
\pagestyle{empty}
\parindent=0em
\parskip=0pt
```

Now calculate the size of labels simply as a proportion of the page size (if you haven't got that right, this won't work, will it?). This macro is to be executed before the first label is made. In environment `labels` and macro `\genericlabel` (see below) this happens automatically. If you write your own label environment or macro, then model it after one of the aforementioned.

```
\def\@LabelSetup{%
\global\label@width\textwidth%
\global\divide\label@width by\LabelCols%
\global\label@height\textheight%
\global\advance\label@height by-\TopBorder%
\global\advance\label@height by-\BottomBorder%
\global\divide\label@height by\LabelRows%
```

The top margin of the paper is generally unused for labels. Avery 5360 label sheets have a 1/4-inch gutter on the top and bottom. However, there are no left or right margin gutters. We adjust the top margin to keep the labels from printing on the gutter.

```
\global\topmargin=-1in\global\advance\topmargin by\TopBorder%
```

It is not usually advisable to make the label printing go right to the edge of the available area, so `\area@width` gives the area that will actually be used for printing; the width is cut down by `\LeftBorder` plus `\RightBorder`. These dimensions can be set to zero if you have a design that uses the whole label.

```
\global\area@width=\label@width%
\global\advance\area@width by -\LeftBorder%
\global\advance\area@width by -\RightBorder%
```

If the labels are to be produced on a grid, for debugging, then the usual setting of `\fbox` separator width and `\fbox` rule width must be subtracted so that the label box doesn't crash into the grid.

```
\ifLabelGrid%
\global\advance\area@width by-2\fboxsep%
\global\advance\area@width by-2\fboxrule%
\fi%
```

The height of the label box will be `2\half@label`. However, this box will definitely crash into the grid lines. Reduce the label height to account for the rule and separator widths that will be added automatically by the boxing routine later on.

```
\ifLabelGrid%
\global\advance\label@height by-2\fboxsep%
\global\advance\label@height by-2\fboxrule%
\fi%
\global\half@label=\label@height\divide\half@label by2\relax
\global\label@number=1\relax
}
```

The boolean variable `\LabelInfotrue` is used to toggle the amount of information printed at runtime. The boolean is placed in the preamble to invoke a more informative printout. The default is no information, but still echo a message, saying how to print more information.

```
\def\LabelSetup{\@LabelSetup%
\ifLabelInfo
   \typeout{Control sequences adjustable in the preamble:}
   \typeout{\LabelRows=\the\LabelRows}
   \typeout{\LabelCols=\the\LabelCols}
   \TypeoutBlurb{\TopBorder}{\TopBorder}
   \TypeoutBlurb{\BottomBorder}{\BottomBorder}
   \TypeoutBlurb{\LeftBorder}{\LeftBorder}
   \TypeoutBlurb{\RightBorder}{\RightBorder}
   \typeout{Computed values:}
   \TypeoutBlurb{Label Width}{\label@width}
   \TypeoutBlurb{Label Height}{\label@height}
\else
   \typeout{To print info, put ‘\protect\LabelInfotrue’ in the preamble}
\fi
}
```

We might want to print the same label several times, so `\sticky@label` will repeat `\make@label` a specified number of times (`\numberoflabels`)

```
\numberoflabels=1%
```

```
\def\sticky@label{\l@so@far=0%
\loop\ifnum\l@so@far<\numberoflabels\advance\l@so@far by 1\make@label%
\repeat}
```

The real label-making macro appears below. It assumes the actual text is in a box called `\this@label`. It is vital to make sure spaces are not included at the end of lines in these macros, or all hell breaks loose. Internal checks are made for box width and height, to report violations. The environments and macros provided below always produce a ragged right box of fixed width. Use `\genericlabel` to defeat the ragged right box and fixed box width.

```
\def\make@label{%
\ifLabelGrid%
 \let\boxing@type\framebox%
\else%
 \let\boxing@type\makebox%
\fi%
```

The boxes made by the method below can overflow horizontally or vertically. The code below emits an error message which pinpoints the trouble and the degree of difficulty encountered. Most of the time the text wrapping and ragged right controls fix the problem, but some extra long lines can be troublesome. An essential part of this code is to test the box for zero width, which will emit an empty label. If we are doing it on purpose, then emit the label, otherwise discard it.

```
\settowidth{\LabTmp}{\usebox{\this@label}}%
\ifdim\LabTmp=0cm\let\action=\relax\else\let\action=\BuildB@x\fi%
%\TypeoutBlurb{Box Width}{\LabTmp}%
\advance\LabTmp by -\area@width%
\ifdim\LabTmp>0cm\relax%
\PkgBlurb{Label too wide}{\LabTmp}\fi%
```

```
\settoheight{\LabTmp}%
{\begin{tabular}{l}\usebox{\this@label}\end{tabular}}%
%\TypeoutBlurb{Box height}{\LabTmp}%
\advance\LabTmp by -\half@label%
\ifdim\LabTmp>0cm\relax%
\LabTmp=2\LabTmp\PkgBlurb{Label too tall}{\LabTmp}\fi%
```

Check for an empty box. Build the box provided it has some dimension or else we are skipping labels on purpose. Otherwise, its empty and we discard it.

```
\ifnum\skip@labels<\sk@pped\let\action=\BuildB@x\fi%
\action
}
```

Now build the box for the actual label. The box has minimum height, which is set by using a vertical rule of zero width. At the same time, set a minimum box width. Set a position to half-way up a strut of the height of the label, thus forcing text to be the correct height and vertically centered. Apply box methods to adjust the white space left and right, using horizontal rules of zero height.

```
\def\BuildB@x{%
\boxing@type[\label@width][c]{%
\rule{0pt}{\label@height}%
\raisebox{\half@label}[0pt][0pt]{%
\rule{\LeftBorder}{0pt}%
\usebox{\this@label}%
\rule{\RightBorder}{0pt}%
}}%
```

Print `+` for this label to target errors by label number. Empty labels made on purpose are printed as `x`. Start a new line and print `|` if we have printed a row of `\LabelCols` labels.

```
\ifnum\skip@labels<\sk@pped\message{x}\else\message{+}\fi%
\ifnum\label@number=\LabelCols%
\message{|}%
\endgraf\nointerlineskip%
\global\label@number=1\else\global\advance\label@number by 1\fi%
}%
```

To print out dimensions in more usual millimeters, the following macros are used, which convert from TEX's scaled points into millimeters. The common TEX points are also printed.

```
\newcount\@Milli
\def\ToMilli#1{\@Milli=#1\advance\@Milli by93225\relax
\divide\@Milli by 186450\relax}
\def\PkgBlurb#1#2{\ToMilli{#2}%
\PackageWarning{labels}{#1 by \the\@Milli mm (\the #2)}%
}
\def\TypeoutBlurb#1#2{\ToMilli{#2}%
\typeout{#1=\the\@Milli mm (\the #2)}%
}
```

3 User macros

The basic case is a generic macro `\genericlabel{#1}` which takes its argument `#1` and puts it out on a label. No ragged right. No fixed width. Very basic.

```
\newcommand{\genericlabel}[1]{%
\iffirst@label\LabelSetup\first@labelfalse\fi%
\savebox{\this@label}{#1}\sticky@label\@gobblecr}
```

The more useful macro `\addresslabel[#1]{#2}` is based upon a tabular environment, therefore it accepts lines ending in `\\`. It is supposed to reproduce what is made by the `labels` environment. Extra space left and right is removed from the tabular environment and the width is fixed at value `\area@width` (see above for a discussion of this computed dimension) with ragged right edge. The optional argument `[#1]` is provided in order to set local values of certain variables, for example, `[\fboxrule=2pt]` could appear as the optional argument in order to locally change box rule size, without affecting the grid line size used for debugging. The plan is to use the common denominator of 2.09 and 2e, so first we define some double argument double-talk.

```
\long\def\L@dblarg#1{\@ifnextchar[{#1}{\L@xdblarg{#1}}}
\long\def\L@xdblarg#1#2{#1[]{#2}}
```

Using LaTeX2e extensions, the command `\addresslabel[#1]{#2}` could be coded as `\newcommand{\addresslabel}[2][]{...}`. In coding common to both 2.09 and 2e, the following works:

```
\long\def\addresslabel{\L@dblarg{\@addresslabel}}
\long\def\@addresslabel[#1]#2{\genericlabel{#1%
\begin{tabular}{@{}p{\area@width}@{}}\raggedright #2\end{tabular}}}
```

The macro `\boxedaddresslabel[#1]{#2}` adds a framed box around the address label defined above. The trouble here is in determining the box width, which is reduced from its normal size by the widths of the box rule and rule separator. The optional argument `[#1]` is the same as for the previous macro, but here it performs a real service, because it is often the case that box rule and box separator sizes should be adjusted locally.

```
\long\def\boxedaddresslabel{\L@dblarg{\@boxedaddresslabel}}
\long\def\@boxedaddresslabel[#1]#2{\genericlabel{#1%
\addr@width=\area@width\advance\addr@width by-2\fboxsep%
\advance\addr@width by-2\fboxrule\fbox{%
\begin{tabular}{@{}p{\addr@width}@{}}\raggedright #2\end{tabular}}}}
```

The environment `labels`, for verbatim labels, will be defined.

```
\long\def\labels
{\iffirst@label\LabelSetup\first@labelfalse\fi\start@@label}
\def\endlabels{\end@@label}
```

The label contents are saved in a box called `\this@label`, formed as a raggedright minipage of width `\area@width`. The trick is to make the end of line character into a macro `^^M` which gets executed each time it is encountered. By testing for blank lines, we can find the end of a series of consecutive address lines and cause the macro to terminate the label box definition, and possibly start another. The usual action taken on intercept of an end of line character is to insert `\newline`. It is only the case of a blank line that causes us to end the current label. Here, swallow a pending `^M`, to avoid having a blank line at the start of each label. Extra blank lines cause blank labels, which are thrown away when encountered, later on.

```
\def\start@@label{%
\savebox{\this@label}\bgroup\raggedright%
\begin{minipage}{\area@width}\raggedright%
\catcode `\^^M =\active\@gobblecr%
```

```
}%
\def\end@@label{%
\end{minipage}\egroup%
\sticky@label}%
```

The principal support macros needed to define the `labels` environment will be defined below. These macros assume names and addresses appear as consecutive lines separated by a blank line. If we are in the middle of consecutive address lines, then just start a new line.

```
\def\start@newline{\mbox{}\expandafter\newline}%
```

If we have met a blank line, then finish the current label and start a new label.

```
\def\new@label{\end@@label\start@@label}%
```

The macro `^^M` invokes either `\start@newline` or `\new@label`. To define it, we use some hackery from Phil Taylor. Step one is to activate a control sequence at end of line.

```
\catcode `\^^M = \active%
```

The opaque definition required uses `futurelet` to selectively expand a control sequence during a definition. The macro `^^M` is supposed to replace carriage returns by `\newline`, or else end this label and start a new one with `\new@label`. The idea is to define `^^M` to be a control sequence `\nexttoken`, active only when `^^M` has been made an active control sequence. Then the expansion of `^^M{#1}` will be `\nexttoken`, provided the next line `#1` following the current one has no characters (except carriage return at the end). Otherwise, we must be at the start of a new cluster of label lines.

```
\def ^^M{\futurelet\nexttoken\isitapar}%
\def\isitapar{\ifx^^M\nexttoken\let\action=\new@label\else%
\let\action\start@newline\fi\action}%
```

Definitions involving `^^M` are finished. Re-instate the original catcode for carriage-return.

```
\catcode `\^^M = 5\relax%
```

The syntax of the `labels` environment can be shortened to a single line, provided an input file name is known and the contents have been prepared for use with the `labels` environment.

```
\def\labelfile#1{\begin{labels}\input#1\end{labels}}
```

In the interest of a general engine for label production, the program can prompt for the file name at runtime. This kind of feature makes it possible to run one batch file to create labels, regardless of the source.

```
\def\promptlabels{\typein[\labelfilename]{What is the name of the
label file?}
\labelfile{\labelfilename}}
```

It is possible that a number of empty labels should be printed before going on to print the next set of labels. The `\skiplabels` macro takes an integer argument of the number of empty labels to produce. An `x` is printed on the terminal for each such empty label. For example:

```
\documentclass{article}
\usepackage{labels}
\begin{document}
\numberoflabels=19
```

```
  \boxedaddresslabel{Me \\my street \\ mytown \\ England}
  \skiplabels{2}% Make 21 on first sheet
\numberoflabels=16
  \boxedaddresslabel{You \\your street \\ yourtown \\ England}
\end{document}
```

```
\newcount\sk@pped
\def\skiplabels#1{\sk@pped=#1%
\savebox{\this@label}{\rule{0pt}{.5in}}%
\skip@labels=0\relax
\loop\ifnum\skip@labels<\sk@pped\make@label%
\advance\skip@labels by 1\relax\repeat}
⟨/package⟩
```

4 History and acknowledgements

- (SPQR) v.1 May 9th 1989 simply allowed for `\addresslabel{... \\ ...\\...}`
- (SPQR) v.2 July 15th permitted verbatim style with no explicit end of lines
- (SPQR) v.3 March 1991 made more generic
- (SPQR) v.4 January 1992 checked and made to work with emtex drivers to my satisfaction, and documented to bare-bones level with 'doc' system.
- (SPQR) v.5 March 1993 allowed for `\skiplabels`
- (SPQR) v.6 January 1994 for LaTeX 2_ε
- (SPQR) v.7 January 1994 fixes
- (SPQR) v.8 April 1994 revised .dtx file
- (GG) v.9 April 1995 revised .dtx file to Package so it works with `article`, `report` and `book` classes. Revisions by `gustafson@math.utah.edu`. Made source independent of paper size (it depended on a4 paper). Fixed extra space bug in tabular array of `\addresslabel` - it did not reproduce results of the `labels` environment. Minipage lines weren't ragged, now they are. New controls to set spaces around labels. Corrected the label placement computations. Error reporting for labels that are too big for the set dimensions. Report in millimeters. Invoked label setup at runtime. New `\boxedaddresslabel[#1]{#2}`. Micrometer placement of label grid on the page. Introduced empty label algorithm. Got rid of strange error messages.
- (GG) v.9 May 1995. `newenvironment{labels}` failed under LaTeX 2.09 (1991), but worked under later versions! Defining `\labels` and `\endlabels` fixed the problem.
- (SPQR) v.10 June 1995. Checked and issued.

The crucial macros which make the system bearable for mailing lists by redefining end of line came from Phil Taylor; apologies to him for using them in a LaTeX style file!