PostScript and M Team Up for Graphical Use

by Christine Bowman

Abstract

Integrating M with the PostScript programming language is an effective way to generate printed output merging data, text, graphic images, and sampled images. The technique is simple and the results are powerful. By imbedding PostScript code into M code, M data can be merged with or represented as graphics objects. Potential applications include generating invoices with customer information, mailing labels, calendars, certificates, diplomas, preprinted checks, text merged with images, and a variety of other graphically based output. Integrating PostScript code with M data reduces or eliminates the need for preprinted forms, invoices, labels, and logos. By representing M data as graphics objects, printed output which includes text as well as eye-catching graphics can easily and efficiently be generated.

Introduction

PostScript is a simple but powerful page-definition, programming language useful in displaying graphics on the printed page. The language contains a wide range of graphics operators; it allows the creation and manipulation of variables, supports procedure and function definition, and includes a conventional set of data types and control primitives.[1,2] M has the ability to manage large databases and efficiently manipulate character-based data, has a powerful command structure, and uses highly efficient data-storage techniques. By combining these two applications, one can integrate M data with PostScript graphics operators to produce a variety of printed-page formats.

Typically, PostScript applications are created and used in the workstation environment. There are, however, no restrictions to writing customized PostScript code for applications existing on a mainframe system, thereby enabling PostScript to be integrated with the existing database and applications. Workstation (desktop and graphics) applications generate PostScript code to produce the desired output. A programmer can write efficient and compact PostScript code to deliver the desired output directly from the main system without needing workstation application software. Putting the PostScript application on the main system offers the advantages of directly merging data with graphics output, implementing efficient and compact code, and sharing PostScript printers among many applications and users.

PostScript programs are written in ASCII text, and can be created using an ordinary text editor. There is no compiled or encoded form of the language; the code is device-independent, meaning that code that works on one model of PostScript printer will work on any other standard PostScript printer, since interpretation of PostScript code takes place within the printer controller.[3,4,5] In the applications described in this article, M programs generate text files containing PostScript code and M data. When printed on a PostScript printer, these files are interpreted via the language interpreter residing within the PostScript printer to produce the desired drawing, form, or graphic.

Methodology

The applications described here were developed using a VAX/VMS system running DSM V5.2, VA Kernel V6.5, and Digital Equipment Corporation's ScriptPrinter software V1.1. Various printers were used, including a DEC LNO3R, Okidata 840 Laser Printer, Hewlett Packard LaserJet IIID, and an Apple LaserWriter printer.[6] VMS print queues were set up to support the printers in accordance with instructions in the ScriptPrinter manual for a PostScript print queue and an ANSI print queue.[7] Each printer had a minimum of 4 MB RAM. The applications were developed using M code, and generally run as a menu option under the VA MenuManager system.

The Mailing Labels

The mailing labels shown in figure 1 consist of lines, fixed text (such as "SHIP TO:"), a company logo, and data from the database, which includes names and addresses. The M programming steps used to generate these labels are:

- M `OPEN` and M `USE` statements to open and write to a text file;
- M `WRITE` statements to write PostScript code to the opened text file;
- M code to select names and addresses from the database and write them to the text file; and
- M `CLOSE` statement to close and print the text file to a PostScript print queue.

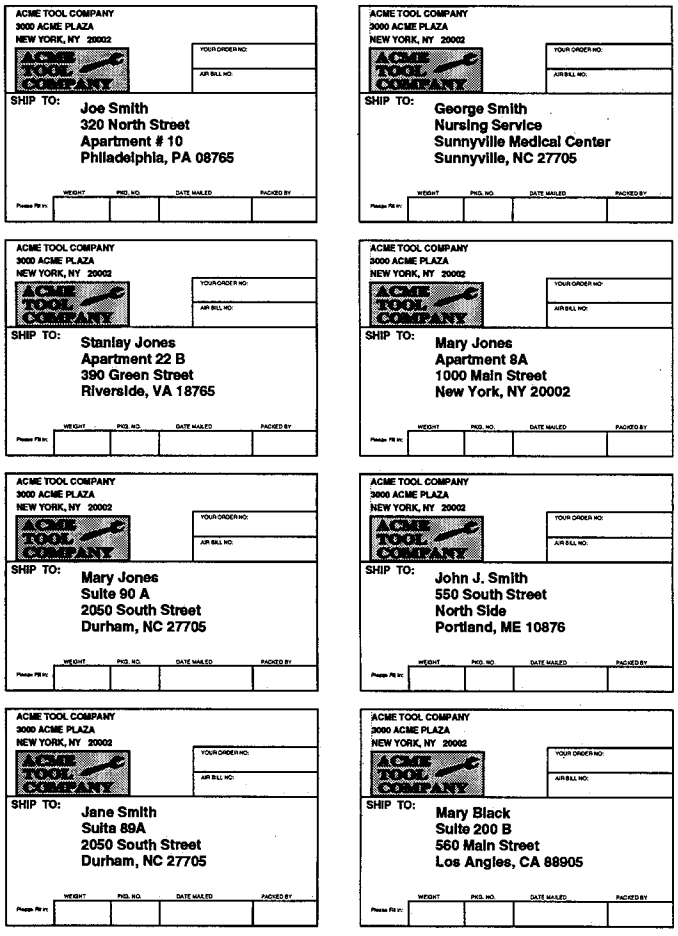

Figure 1. Mailing labels generated by M data.

A more complete listing of the code is given in appendix A. The M routine "label" creates a text file named "PSFILE" that is printed to a PostScript printer. Lines PSTART through PSTOP are PostScript code, lines NAME through END are M code to extract name information from the database. Line END+1 prints the text file to a PostScript print queue. A complete listing of the text file is given in appendix B. Comments, denoted by %%, have been added for clarity, but will not appear in the text file.

The PostScript syntax "/keyname {operations}def" defines a PostScript procedure. When the PostScript interpreter encounters the literal name "keyname", it executes the associated procedure. In this example, PostScript procedures called "form", "logo", and "dolabel" are executed for each label printed.

Graphics descriptions, such as the mailing label shown here, can be scaled, rotated, and translated (in terms of x, y coordinates) to form new output. [8] For example, the output shown in figure 1 was printed on blank 3.2" X 2.2" labels (eight labels per page). Modifying the routine to accommodate different label sizes is easily accomplished by using the PostScript "scale" command. In routine "label", line PSTOP–4 shows usage of the scale command. Modifying scale parameters produces a label of any desired size, for example, changing ".8 .8 scale" to ".5 .8 scale" produces a label approximately 2"x2.2" in size. The full size of the label demonstrated in this program is 4" x 2.75" (1:1 scale). Similarly, to change the location or spacing of the labels on the page, the parameters (top margin, bottom margin, space between labels, etc.) defined in lines PSTART+12 through PSTART+15 are easy to modify. Font sizes and styles are modified by making appropriate changes to lines PSTART+8 through PSTART+11.

Printing an Invoice

Figure 2 shows an invoice generated using M. In general, an invoice consists of three types of information: graphics information—for example, logos, lines, check boxes, or text—that is constant on every invoice; field data that are always printed in the same location on the invoice, but vary in content from invoice to invoice (NAME, ADDRESS, DATE); and check marks or other marks on the invoice in various locations. For example, a check mark is placed to indicate a particular method of payment. Field data and "check box" information are found in the database; the invoice form is produced by PostScript code generated by the M routine.

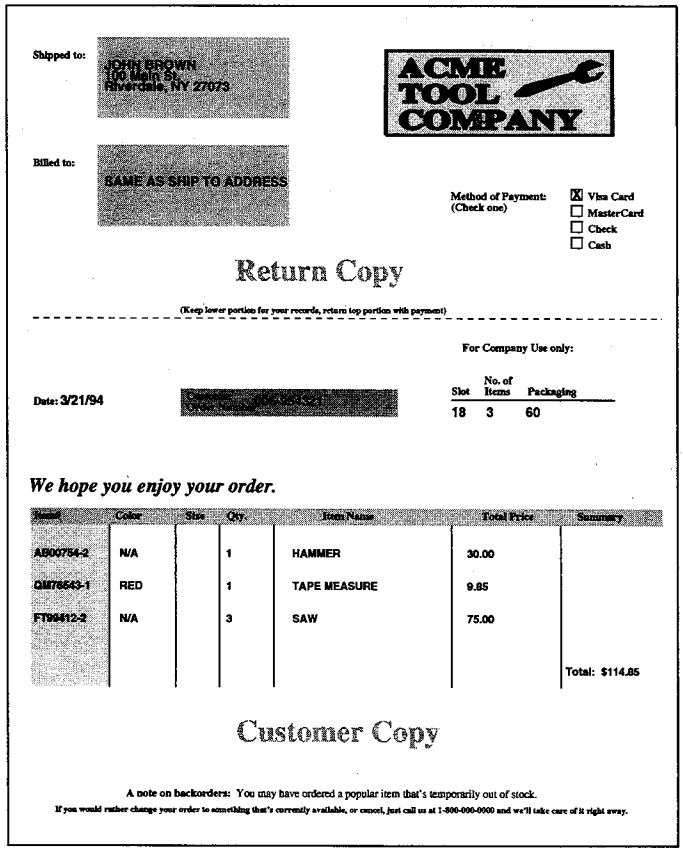

Figure 2. Sample invoice.

To create the printed invoice with data, the M routine creates the text file which contains PostScript code definition of the form. The field data (NAME, DATE, SSN) and check-box information are selected from the database and included in the text file, as was done in the last example. The PostScript form definition contains an additional piece of information that defines which check boxes are filled in on the individual form according to data supplied by the database. The PostScript definitions of payment methods are defined in terms of "moveto" statements. This PostScript code is included as part of the form definition. Payment method is selected from the database and written to the text file; in this example, the word "VISA" is written to the file. When the PostScript interpreter encounters the text VISA, it executes the procedure named VISA, as the code segment in figure 3 demonstrates.

```
%% Define invoice form
/form {PostScript code to draw invoice form shown in figure 2}def
%% Define check box locations in terms of data variables
/VISA {400 600 moveto (X) show}def
/Master {400 590 moveto (X) show}def
/Check {400 580 moveto (X) show}def
/Cash {400 570 moveto (X) show}def
%% Draw invoices and place specific customer data on form
form        %% draws invoice form
50 710 moveto (JOHN BROWN) show    %% fill in SHIP TO: information
50 690 moveto (100 Main St.) show
50 670 moveto (Riverdale, NY   27073) show
VISA      %% put an X in the VISA box
```

Figure 3. Code segment for indicating payment method on invoice.

Using M Data as Objects

A final example demonstrates how information can be selected from an M database and output as an object, such as a picture, graphic, or logo. Using PostScript syntax for procedure definition, a literal name can be defined in terms of its graphics operators. The literal name (or field data) is selected from the M database, and sent to the text file, which also includes appropriate PostScript code. When the text file is sent to the PostScript printer, the procedure definition is executed when the literal name is encountered, and the graphic object representing the piece of data is drawn on the printed page.

For example, parking permits are issued and printed using M and PostScript. Employee information tracked in the database includes name, Social Security number, automobile (model), and special parking fields. Special parking might include data such as handicapped or reserved space. The special parking data will print out as a picture on the parking permit rather than as text. A procedure for each special parking data item is created. When printing the permit, the interpreter encounters the field and executes the associated procedure, demonstrated by the following code segment in figure 4. Output of this code is shown in figure 5.

```
%% Procedure to draw the handicapped symbol, mostly lineto commands
/rl {rlineto} def
/handicapped {584 636 moveto 603 567 72 105 330 arc -10 -10 rl 603 567 54 336 105 arcn -5 -18 rl 594 586
moveto 0 105 rl 603 675 18 100 300 arcn 0 -27 rl 40 0 rl 0 -10 rl -40 0 rl 0 -22 rl 50 12 rl 20 -45 rl 13
9 rl 5 -10 rl -26 -18 rl -23 45 rl -10 -10 rl stroke }def
%% Write text on the parking permit
216 432 moveto (NAME: JOHN BROWN) show
216 420 rm (SSN: 123-45-6789) show
216 408 rm (PHONE: 201-999-1212) show
%% draw the handicapped symbol on the permit
handicapped
```

Figure 4. Code segment to print a special parking sticker.

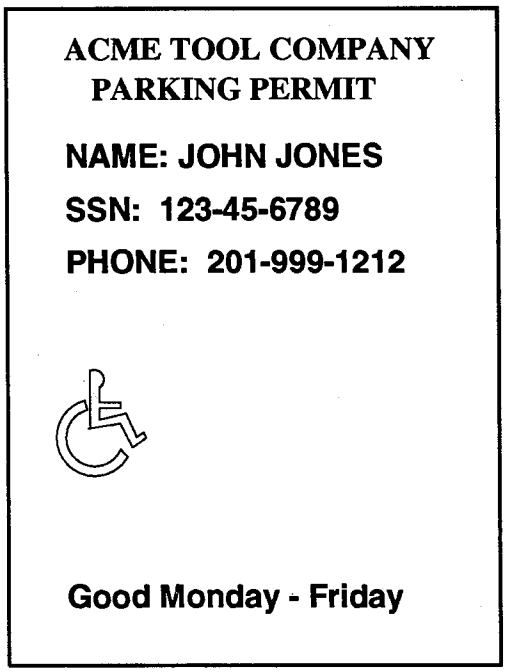

Figure 5. M data can be printed as a graphic symbol.

Other useful examples might include warnings, such as might be used in a medical database and in the printing of patient records. If a drug allergy is present, a high lab value detected, or a similar warning is present, an eye-catching graphical image could easily and automatically be printed on the patient record. Similarly, shipping labels could be generated with standard symbols for hazards included as part of the printed label. Standard indicators such as "Radioactive," "Flammable," "Corrosive," or "Biohazard" symbols are relatively easy to reproduce with PostScript code. The symbol would be part of the data defining the material to be shipped and correct labeling and warning is ensured.

Summary

There are numerous other examples of using PostScript with M. Bitmap images can be included and used as a logo, or scanned signatures, drawings, or sampled images can be input and used with text files. For example, a signature could be scanned and used to print out with M data on a document, license, or certificate. The disadvantage here is the large size of bitmap or image files. When possible, graphics operators should be used for increased efficiency and compact file size.

M data could be aggregated (counts, totals, sums, etc.) and easily represented as a pie chart, bar chart, or other graphical display. The introduction of color PostScript printers adds more features to the variety of applications which could be developed. For simple PostScript designs, such as those demonstrated in this paper, the code can be written directly with M commands. For complex designs or those with many procedure definitions, a text file can be created and appended to the output files using operating system "append" or "merge" commands to avoid having to write lengthy PostScript code segments within the M application.

Conclusion

Integrating PostScript with M code allows one to directly print high-quality graphics output from the M application. M data can be merged with graphics operators to produce a variety of possible output formats. M data can be visually represented as graphics objects, which can be placed side by side with standard text output. While PostScript code can become quite complex, the M programming steps necessary to use this application are relatively simple and straightforward. As a flexible programming environment, PostScript code can be modified, combined, and built on to produce complex pages from simple definitions. PostScript code, such as the examples shown in this paper, can generate useful applications to meet a variety of user needs.

Christine Bowman is a computer specialist at the Department of Veterans Affairs Quality Management Institute in Durham, North Carolina, and is in a graduate training program in medical informatics. She has a degree in mechanical engineering and a master's degree in biomedical engineering.

Endnotes

1. Adobe Systems Incorporated, *PostScript Language Reference Manual* (Reading, MA: Addison-Wesley Press, 1989).
2. Adobe Systems Incorporated, *PostScript Language Tutorial and Cookbook* (Reading, MA: Addison-Wesley Press, 1985).
3. Adobe Systems, *PostScript Language Reference Manual.*
4. Adobe Systems, *Postscript Language Tutorial.*
5. Digital Equipment Corporation, *VAX/VMS Software Installation Guide: LN03R ScriptPrinter* (Digital Equipment Corporation, Maynard, MA, 1988).
6. Digital Equipment Corporation, *LN03R ScriptPrinter Programmer's Supplement* (Digital Equipment Corporation, Maynard MA, 1987).
7. Digital Equipment Corporation, *VAX/VMS.*
8. Adobe Systems, *PostScript Language Tutorial.*

Appendix A. M routine "label" generates mailing labels with logo.

```
label ;cbowman;May 1990
      ;generate mailing labels for printing on PostScript printer
      ;Use laserwriter labels, 2.75" X 4", 8 labels per sheet
      ;
      ; Open text file named PSFILE
      O "PSFILE":NEWVERSION U "PSFILE"
      ; Write PostScript code to the text file
PSTARTW "gsave",!,"/mto {xx yy moveto} def /rl {rlineto}def ",!
      W "/form",!," { mto 288 0 rlineto 0 23 rmoveto -243 0 rlineto",!
      W "0 -23 rlineto 52 0 rmoveto 0 23 rlineto 45 0 rmoveto "
      W "0 -23 rlineto 86 0 rmoveto 0 23 rlineto stroke "
      W "mto 0 200 rlineto 288 0 rlineto 0 -200 rlineto -278 0  rlineto "
      W "stroke xx yy 119 add moveto 288 0 rlineto "
      W "0 23 rmoveto -115 0 rlineto 0 23 rmoveto 0 -46 rlineto"
      W "0 46 rmoveto 115 0 rlineto -176 -28 rmoveto stroke}def"
      W "/sfont {/Helvetica findfont 5 scalefont setfont}def",!
      W "/bfont {/Helvetica-Bold findfont 10 scalefont setfont}def",!
      W "/bbfont {/Helvetica-Bold findfont 13 scalefont setfont}def",!
      W "/tfont {Times-BoldItalic findfont 16 scalefont setfont}def",!
      W "/height 207 def /lpl 3 def /lspace {height lpl div}def",!
      W "/bottom 50 def /right 610 def /left 50 def /top 740 def",!
      W "/lwidth 288 def /colspace 40 def /lwidth colspace lwidth add def",!
      W "/rowspace 10 def /height rowspace add def",!
      W "/textlines {",!
      W "sfont mto 10 13 rmoveto (Please Fill in: ) show ",!
      W "mto 54 25 rmoveto (WEIGHT) show 29 0 rmoveto (PKG. NO.) show",!
      W "32 0 rmoveto (DATE MAILED) show",!
      W "45 0 rmoveto (PACKED BY) show",!
      W "bfont mto 5 108 rmoveto (SHIP TO:) show sfont",!
      W "mto 180 158 rmoveto (YOUR ORDER NO:) show",!
      W "mto 180 135 rmoveto (AIR BILL NO:) show",!
      W "bfont mto 10 190 rmoveto (ACME TOOL COMPANY) show",!
      W "mto 10 178 rmoveto (3000 ACME PLAZA) show",!
      W "mto 10 166 rmoveto (NEW YORK, NY   20002) show",!,"}def",!
      W "bbfont",!,"mto 70 55 rmoveto show",!,"mto 70 70 rmoveto show",!
      W "mto 70 85 rmoveto show mto 70 100 rmoveto show",!,"mto logo grestore"
      W " }def",!,"/logo {",!
      W "gsave mto 260 0 rl 0 130 rl -260 0 rl 0 -130 rl .9 setgray fill "
      W "newpath grestore lfont newpath mto 15 90 rmoveto (ACME) true"
      W "charpath mto 15 50 rmoveto (TOOL) true charpath mto 15 10 "
      W "rmoveto (COMPANY) true charpath gsave 3 setlinewidth stroke"
      W "grestore .5 setgray fill newpath mto 260 0 rl 0 130 rl -260 "
      W "0 rl 0 -130 rl 4 setlinewidth 0 setgray stroke mto 148 60 "
      W "rmoveto 14 0 rl 54 27 rl 10 -7 rl 18 7 rl 5 10 rl -7 -3 rl"
      W "0 -4 rl -9 -1 rl -4 10 rl 20 10 rl -15 7 rl -12 -7 rl -5 -12 rl"
      W "-57 -20 rl -12 -15 rl gsave 1 setlinewidth stroke grestore"
      W " .3 setgray fill}def",!
      W "/dolabel { aload pop ",!
      W "left lwidth right{/xx exch def",!
      W "bottom height top{/yy exch def .8 .8 scale form mto textlines }"
PSTOP W"for /yy bottom def}for",!,"showpage ",!,"}def",!!,"["
      ;Get names and addresses from database
      S CTR=0
NAMES ;
      D ^GETNAME
ADDR  D ^ADDRGET
      ;Routines GETNAME and ADDRGET put name and address information
      ;into the variables NAME=first and last name to be printed on label
      ;STR1= Street Address line 1
      ;STR2= Street Address line 2
      ;CIST= City, State, ZIP Code concatenated together
      W "("_NAME_")",!,"("_STR1_")",!,"("_STR2_")",!,"("_CIST_")",!
      S CTR=CTR+1
      I CTR=8 W "]",!,"dolabel",!,G NAMES
      I NAME=-1 G END
      G NAMES
END   ;
      C "PSFILE":(SPOOL:QUEUE="PSCRIPT")
      K NAME,STR1,STR2,CIST,CTR
      Q
```

Appendix B. Textfile created by routine "labels".

```
%% all measurements are in points, one point=1/72 inch.

%% line drawing: draws all necessary lines on the label
gsave
/mto {xx yy moveto} def /rl {rlineto}def
/form
{ mto 288 0 rlineto 0 23 rmoveto -243 0 rlineto
0 -23 rlineto 52  0 rmoveto 0 23 rlineto 45 0 rmoveto
0 -23 rlineto 86 0 rmoveto 0 23 rlineto stroke
mto 0 200 rline to 288 0 rlineto 0 -200 rlineto -278 0 rlineto
stroke xx yy 119 add moveto 288 0 rlineto
0 23 rmoveto -115 0 rlineto 0 23 rmoveto 0 -46 rlineto
0 46 rmoveto 115 0 rlineto -176 -28 rmoveto stroke}def

%% define some fonts and variables to be used on the label
/sfont {/Helvetica findfont 5 scalefont setfont}def
/bfont {/Helvetica-Bold findfont 10 scalefont setfont}def
/bbfont {/Helvetica-Bold findfont 13 scalefont setfont}def
/tfont  {Times-BoldItalic findfont 16 scalefont setfont}def
/height 207 def /lpl 3 def /lspace {height lpl div}def
/bottom 50 def /right 610 def /left 50 def /top 740 def
/lwidth 288 def /colspace 40 def /lwidth colspace lwidth add def
/rowspace 10 def /height rowspace add def

%% fixed text: draws all fixed text on the label
/textlines {
sfont mto 10 13 rmoveto (Please Fill in: ) show
mto 54 23 rmoveto (WEIGHT) show 29 0 rmoveto (PKG. NO.) show
32 0 rmoveto (DATE MAILED) show
45 0 rmoveto (PACKED BY) show
bfont mto 5 108 rmoveto (SHIP TO:) show sfont
mto 180 158 rmoveto (YOUR ORDER NO:) show
mto 180 135 rmoveto (AIR BILL NO:) show
bfont mto 10 190 rmoveto (ACME TOOL COMPANY) show
mto 10 178 rmoveto (3000 ACME PLAZA) show
mto 10 166 rmoveto (NEW YORK, NY 20002) show
bbfont
mto 70 55 rmoveto show
mto 70 70 rmoveto show
mto 70 85 rmoveto show
mot 70 100 rmoveto show
mto logo grestore
}def

%% ACME logo on mailing label
/logo {
gsave mto 260 0 rl 0 130 rl -260 0 rl 0 -130 rl .9 setgray fill
newpath grestore lfont newpath mto 15 90 rmoveto (ACME) true
charpath mto 15 50 rmoveto (TOOL) true charpath mto 15 10
rmoveto (COMPANY) true charpath gsave 3 setlinewidth stroke
grestore .5 setgray fill newpath mto 260 0 rl 0 130 rl -260
0 rl 0 -130 rl 4 setlinewidth 0 setgray stroke mto 148 60
rmoveto 14 0 rl 54 27 rl 10 -7 rl 18 7 rl 5 10 rl -7 -3 rl
0 -4 rl -9 -1 rl -4 -10 rl 20 10 rl -15 7 rl -12 -7 rl -5 -12 rl
-57 -20 rl -12 -15 rl gsave 1 setlinewidth stroke grestore
.3 setgray fill}def
}def

%% dolabel does everything, lines, text, logo, and loads names on stack
/dolabel { aload pop
left lwidth right{/xx exch def
bottom height top{/yy exch def .8 .8. scale form mto textlines }for /yy bottom def}for
showpage
}def
```

```
%% data extracted from the M database
[(George Smith)
(Nursing Service)
(Sunnyville Medical Center)
(Sunnyville, NC 27705)
(Mary Jones)
(Apartment 8A)
(1000 Main Street)
(New York, NY 20002)
(John J. Smith)
(550 South Street)
(North Side)
(Portland, ME 10876)
(Mary Black)
(Suite 200 B)
(560 Main Street)
(Los Angles, CA 88905)
(Joe Smith)
(320 North Street)
(Apartment # 10)
(Philadelphia, PA 08765)
(Stanley Jones)
(Apartment 22 B)
(390 Green Street)
(Riverside, VA 18765)
(Mary Jones)
(Suite 90 A)
(2050 South Street)
(Durham, NC 27705)
(Jane Smith)
(Suite 89A)
(2050 South Street)
(Durham, NC 27705)
]
dolabel
```