Forms and More:
Getting What You Want from Your Printer

Frank Roediger, Meridian Software, Inc.

ABSTRACT

The biggest holdup to producing a final report is sometimes the printing of it: all your number-crunching code works fine but you cannot get the printer to produce a hardcopy that matches the detailed report specs that a finicky user has given you. When this happens, Release 6.12 of SAS® for PC platforms like Windows NT® gives you a great deal of control in customizing how a document gets printed. However, if you are working on a UNIX system, many of the PC-based printer control features are not available to you. There, you need to rely on the FORMS subsystem to control printer settings. Regardless of which platform you use, there are also situations such as recovering from printer jams and printing special characters (such as *plus-over-minus* (±) that require special attention. This paper will walk you through these and other printing challenges and present solutions that can work for both UNIX and Windows NT platforms.

INTRODUCTION

As a SAS programmer, your primary focus in producing a report is most often at the data level, setting up a series of PROC and DATA steps and assembling their outputs into a coherent presentation of numbers and labels. For your end-users, however, the primary focus is most often the presentation of the results of your number churning so that their audience can readily see how the data supports the conclusions they have drawn. From the end-users' point-of-view, your data-level number churning, no matter how complicated, is a given; their biggest concern is how the results are presented, not on what you did to create them. This difference in perspective means that just when you think you have finished a report program, the task of refining its final appearance has just begun.

The most straightforward way to control the appearance of printed output is to use the facilities that SAS makes available to you with DMS. The ***Program Editor***, ***Output***, and ***Log*** windows all provide you with direct access to the printer control capabilities. Click on File...⇨Print... and SAS presents you with a series of windows to control the character font and size, the margins, and the orientation of your output. That approach is fine as far as it goes, but it does have its limitations.

§ Not all platforms give you access to this capability: for instance, Windows NT does but on UNIX, File...⇨Print... sends the contents of the active window to the printer.

§ You are limited if what you really need is a set of standard output configurations – you support several user groups and each has specific (and, of course, different) requirements for its report layouts. Any changes you make to the settings via those printer control windows replace what used to be there. To use a particular configuration again, you would need to make sure that all its settings are restored.

SAS provides you with a way of overcoming these limitations with its FORMS subsystem.

The FORMS subsystem is a utility within SAS that allows you to create form definitions. These form definitions act as an intermediary between the contents of a DMS window and the printer, adding printer instructions to the source so that the printer can transform the original into a hard-copy document that is formatted according to your specifications.

ACCESSING THE FORMS SUBSYSTEM

You use the FORMS subsystem to set up SAS catalog entries (generally with a type of FORM and usually located in your SASUSER.PROFILE) that define a set of document characteristics.

If you are using UNIX, a convenient way to access the FORMS subsystem is to click on File...⇨Print Utilities...⇨Open a form.... This displays a window that provides you with data entry fields for libname, catalog name, and form name. If you want the form to be stored in SASUSER, you do not need to fill in the libname entry field; if you want it stored in PROFILE, you do not need to fill in the catalog entry field.

The window also has selection objects for designating whether SAS should assign the form default status. Until a form has been fully developed and tested, it is best not to make it the default (see *USING FORMS*).

This P-Menu access to the FORMS subsystem is not available under Windows NT. There, your only option is to use the FSFORM command from the command window (FSFORMS *form* – remember to include LIBNAME and CATALOG in *form* if the form definition is not going to be an entry in SASUSER.PROFILE). Either way, the FORMS subsystem provides you with access to the following series of screens, each of which controls a set of features for the form you are defining.

Click on Locals to call up a set of options to help you move through the screens.

SETTING UP A FORM

Because of idiosyncrasies among printers and differences in the implementation of SAS across platforms, the process of setting up a form is not generic. To give you a feeling for some of the differences you might encounter when you set up a form, this section walks you through the steps to define a form for two different printers in two different settings. These two examples can serve as a starting point, but you will probably need to make adjustments to their settings before either will fit your needs.

At the bottom of each section are two sets of settings that apply to that screen:

§ the first (UNIX) defines SASUSER.PROFILE.QLAND150 for a QMS 2425 printer using SAS Release 6.12 under HP-UX 10.0

§ the second (NT) defines CLIENT1.PROFILE2.HPORT100 for an HP LaserJet 4L printer using SAS Release 6.12 under Windows NT 4.0.

Printer Selection Window

The ***Printer Selection*** window presents a list of printers. Scroll through the list and press Enter to choose the one you want.

The list may or may not contain the printer you are trying to define a form for; if it does not, choose as an alternate the printer whose description sounds most like the one you will be using. The printer you select in this screen establishes the initial settings that will be made in later screens. Because of this connection between screens, once you have selected a printer, the FORMS subsystem automatically takes you to the next screen **and will not let you go back to this screen!** Therefore, if your printer is not predefined on the list and you have to use an alternate, be prepared to define several trial forms before you get one that matches up with the characteristics of the printer you are using.

TIP: Because you cannot return to this window, you have no way of checking within an existing form to find out which printer it applies to. For this reason, it is helpful to encode the printer designation in the form name or in the description of its catalog entry. (The default description is not too informative -- merely *formname*.FORM.)

To create a new description, you need to be in the FORMS subsystem; on the command line, issue the DES command followed by a quoted string that contains the new form description.

UNIX: Hewlett-Packard LaserJet (+) Courier 10

Note: even though the form is to be used with a QMS printer, QMS is not available in the list provided by the screen. The first choice for an alternate was the first Hewlett-Packard selection on the list because both QMS and Hewlett-Packard printers are PCL-based. As luck would have it, the first choice was an acceptable alternate.

On the command line, enter:

DES ‘QMS 2425, Portrait: ls=150, ps=37’

NT: HP LaserJet Compressed print

On the command line, enter:

DES ‘HP 4L, Portrait: ls=100, ps=55’

Screen 1: Text Body and Margin Information

Text Body: Contains defaults for characters per line, lines on the first page, and lines on following pages. Tab from field to field and override any of the default settings you want to change.

Margins: Contains defaults for left, top, and bottom margins for the first page and for following pages (the right margin is implied from the left margin and the characters per line).

Unless you have output where the first page is printed on a special form such as letterhead, the settings for both the first and the following pages are generally the same. However, the carriage control setting required to get some printers to match up physical pages (the printed sheet) with logical pages (where SAS forces page breaks, based on the current internal setting for PAGESIZE) bumps the first page down a line or two. When this happens, you can give both pages a similar appearance by changing the top margin for the following pages so that their first line prints out the same distance down from the top as the first page’s.

Note that the margins are denominated as characters (horizontal) or lines (vertical), so their width in inches depends upon the pitch and height of your type (which you designate in Screen 5: Printer Control Language). Be prepared to try several different settings before you get a page layout with the border dimensions you desire.

UNIX: Characters per line: 150

Lines on first page: 37

Lines on following: 37

First page Left: 10 Top: 4 Bottom: 0

Following pages Left: 10 Top: 5 Bottom: 0

NT: Characters per line: 100

Lines on first page: 55

Lines on following: 55

First page Left: 21 Top: 0 Bottom: 1

Following pages Left: 21 Top: 2 Bottom: 1

Screen 2: Carriage Control

Generate Carriage Control Information: YES/NO. The documentation is vague about the purpose of this feature: I always use the default of YES.

Signal Page Skips before: Contains a list of situations where SAS can transmit a Form Feed instruction to the printer. Click on a line to toggle it on/off.

On different printers, the use of these can have different results. Used effectively, they make sure that the sheets printed out by the printer match up with the page breaks established in the output by SAS. If they are turned on when they are not needed, blank or partially printed pages can result. A good starting point is to select PRINTER INITIALIZATION and FOLLOWING TEXT PAGES (note that PRINTER INITIALIZATION was not needed for the UNIX/QMS form) .

UNIX: Generate Carriage Control Information: YES

Signal Page Skips before:

Following Text Pages

NT: Generate Carriage Control Information: YES

Signal Page Skips before:

Printer Initialization

Following Text Pages

Screen 3: Print File Parameters

The features of this screen depend upon the host system you are using and the printer you selected in the first screen.

UNIX: Copies: 1

NT: ANSI character set

Screen 4: Font Control Information

The top part of this screen enables you to define characters to act as stand-ins for printer control characters, such as ESC and CTRL. These two characters are common features of PCL commands but are awkward to use – no matter how many times you press the CTRL key, you will not get the CTRL character to be entered on the line you are editing! The way around this dilemma is to assign actual characters that are rarely used to serve as stand-ins for ESC and CTRL. When you want to type the ESC or CTRL character, you type their stand-in character instead.

SAS assigns the tilde (~) as the stand-in for ESC and the caret (^) as the stand-in for CTRL. You cannot change either of these assignments, but you can define as many as six additional ones. You use these stand-ins to define text attribute commands on the bottom part of this screen and PCL commands that you define in the next screen for Printer Control Language.

The bottom part of this screen is for documenting text attribute features, like underlining, boldface, and italics. SAS provides you with an initial set of text attributes, based on the printer you selected in the Printer Selection Window. These text attributes are arranged in pairs:

§ the first item is a sequence of characters that the printer interprets as a single instruction to activate the text attribute

§ the second item is a similar instruction that the printer interprets as a separate instruction to de-activate it.

All the text within the bounds of these instructions will be printed with the designated attribute.

The instructions for turning on and off text attributes are inherent in your printer's on-board operating system. Consult your printer's technical documentation to learn what they are. You use them by embedding them in the text of the report (see *SUGGESTIONS FOR EMBEDDING SPECIAL CHARACTERS*).

UNIX: defaults

NT: defaults

Screen 5: Printer Control Language

This last panel allows you to use Printer Control Language (PCL) to identify any print features that remain. Typically, these include orientation (portrait/landscape), symbol set, font, and character size. These document-wide features are defined in an instruction string under the keyword heading, PRINT INIT.

You can also define an instruction string that is executed before the printing of a specific page by including it under the keyword heading, PAGE *n* (*n* is the page number).

After a print job finishes, it should restore the printer to its default settings with an instruction string under the keyword heading, PRINT TERM.

PCL Tables are generally included as technical appendices in printer manuals. The following two examples include samples of some common PCL instructions.

UNIX: PRINT INIT

~&l1O~(10U~(s0p16.67h8.5v0s0b0T

~&l1O - landscape orientation

~(10U - PC-8 symbol set

~(s0p - initial ESC seq and fixed spacing

16.67h - number of characters/inch

8.5v - primary height

0s - upright (solid)

0b - medium stroke weight

0T - line printer type face

PRINT TERM

~E

~E - reset

NT: PRINT INIT

~&l0O~(8U4102T

~&l0O - portrait orientation

~(8U - Roman-8 symbol set

4102T - Letter Gothic Typeface

PRINT TERM

~E

~E - reset

IMPORTANT NOTE: the italicized text in the above example is there only to explain the components of each ESC sequence. Even though it is good programming practice to have in-line documentation of your code (and these sequences can certainly benefit from being documented), **do not key anything except the keywords and PCL commands into the Printer Control Language Screen.** Any documentation you do for the contents of this screen needs to be done elsewhere.

After you have entered all the information into these screens, you save the form and exit the FORMS Subsystem by clicking on File…⇨ End.

The UNIX form in the above example is the catalog entry, SASUSER.PROFILE.QLAND150.FORM.

The Windows NT form is now the catalog entry, CLIENT1.PROFILE2.HPORT100.FORM.

Because testing and defining a form can involve a great deal of trial-and-error refinements, be prepared for the need to tweak the form's features. To edit an existing form, you invoke the FORMS subsystem the same way you did to define the form. Another way to edit an existing form is to double-click on its entry in the ***Catalog*** window, which you can reach via the Libraries Icon.

SUGGESTIONS FOR EMBEDDING SPECIAL CHARACTERS

For the HP LaserJet 4L, '~&dD' is the escape sequence that turns on italics. However, it is awkward to use this string in a program (e.g., to italicize a word in a title). A convenient way around this is to define the entire escape sequence as a macro variable. This same approach is an effective way of **typing** nonkeyboard characters such as *plus-over-minus* (±) and *greater-than-or-equal-to* (≥). The following DATA step illustrates a method for defining the start and stop instructions for italicizing text and the *plus-over-minus* character as macro variables and using them in a TITLE statement.

```
data _null_;
   length spec_str $200
          spec_chr $1;

   *** Italic - ESC ( s 1 S ******************;
   spec_str=byte(27) ||   /* ESC is 27 (dec) */
            '(s1S';
   call symput('it_strt',trim(left(spec_str)));

   *** Upright - ESC ( s 0 S *****************;
   spec_str='1b'x    ||   /* ESC is 1b (hex) */
            '(s0S';
   call symput('it_stop',trim(left(spec_str)));

   *** in PC-8 character set, plus-over-minus ;
   *** is 241 (dec) / f1 (hex)               ;
   spec_chr='f1'x;
   call symput('specpom',spec_chr);

run;

title "Concentrations of &it_strt.E. Coli&it_stop Within &specpom .05%";
```

One drawback of using this approach is that the resolution of these macro variables produces some peculiar looking results in your ***Output*** window. The start/stop italics commands show up as 5-character strings, with the ESC character being represented by a box. The plus-over-minus character is not part of the character set (Windows 3.0 Latin 1) that SAS uses to display the contents of DMS windows and appears as a peculiar-looking symbol called *lowercase thorn*.

Caution: This same type of misprint will appear on your printout if you do not define the appropriate symbol set in the Printer Control Language screen of your forms definition.

Another drawback is that SAS counts each of the characters in the start-stop italics commands as a printable character when it determines where to center the title. As a result, SAS positions the title to the left of center.

USING FORMS

After you have defined a form, you need to assign it as current before SAS can use its instructions to control printer operations. There are several ways to do this **(NOTE: the first two are not available on Windows NT)**.

§ designate the form as default. When you enter the FORMS subsystem by clicking File...⇨Print Utilities...⇨Open a form... , you are given the opportunity to designate the form you are editing as the default. Assigning a form as the default automatically designates it as the current form at the startup of your SAS session. Having a default form assignment is a good way to safeguard against printing unformatted output (actually, the output will not be unformatted – its formatting is done by whatever settings are active on the printer without any intervention by SAS).

§ Use the File pull-down options: click on File...⇨Print Utilities...⇨Set form name ... , key in the form's name, and click OK.

§ Use the OPTIONS statement: OPTIONS FORMS=*form*

§ Assign a specially designated key: click on Help…⇨ Keys... and define a key with the command, FORMNAME=*form*. Pressing this defined key makes the form current and displays a notification message in the active DMS window.

No matter which option you use, the *form-name* needs to include LIBNAME and CATALOG if it is not an entry in SASUSER.PROFILE.

In Windows NT, there is one additional step: click on File…⇨Print…⇨Setup (or File…⇨Print Setup…) and make sure that the **Use Forms Box** is checked.

Once you have gone through the effort of defining and testing a form, it might be tempting to try and apply it to another printer in a different setting: a form definition is a catalog entry and catalog entries can be copied from system to system. Although transferring forms definitions is feasible, it is not advisable. Every form definition contains printer-specific assignments based on the printer you chose in the Printer Selection Screen. Because you cannot change the printer assignment in an existing form, the transported version of the form definition would need to retain the original's printer-specific features.

LIMITATIONS OF USING FORMS

Forms give UNIX users the same flexibility in printer control that Windows NT users have. They also enable SAS programmers, regardless of which platform they are using, to store printer configurations so that they can conveniently produce reports that conform to a variety of specifications. However, forms, too, have several limitations.

§ They only work for the contents of DMS windows: if you store your reports online, you would first need to open them into the ***Program Editor***. If you do not store reports, you would first need to execute the source code that creates them.

§ They do not allow you to print special characters that are outside the form's native character set (e.g., Windows 3.0 Latin 1 has *plus-over-minus*, but not *greater-than-or-equal*). If you need to print both of these characters, you could use a different character set (e.g., PC-8) that does contain both of the special characters you need. However, changing to a different character set is not always possible: the ones that contain the characters you need may not support the font you are required to use. When you encounter this situation, you need to be able to use more than one character set while printing a single document.

§ They do not allow you to recover from a paper jam.

To overcome these limitations requires going beyond forms.

STORING REPORTS ONLINE

The first step in going beyond forms is to free yourself from having to depend on DMS windows to get your report output to the printer. This entails storing reports online, which involves creating a print file with a FormFeed character embedded at the beginning of every page -- except the first one (the FormFeed characters are vital because they signal to the printer when it has received the contents of a page). There are several ways that you can use SAS to build a report file with embedded carriage control:

§ using a DATA _NULL_ step that contains a FILE statement with the PRINT option

§ using a PROC step whose output has been re-directed to a file by PROC PRINTTO

§ saving the contents of the ***Output*** window after identifying its destination by clicking on File…⇨Print Utilities…⇨Set Print File… (this approach is not available to users of Windows NT).

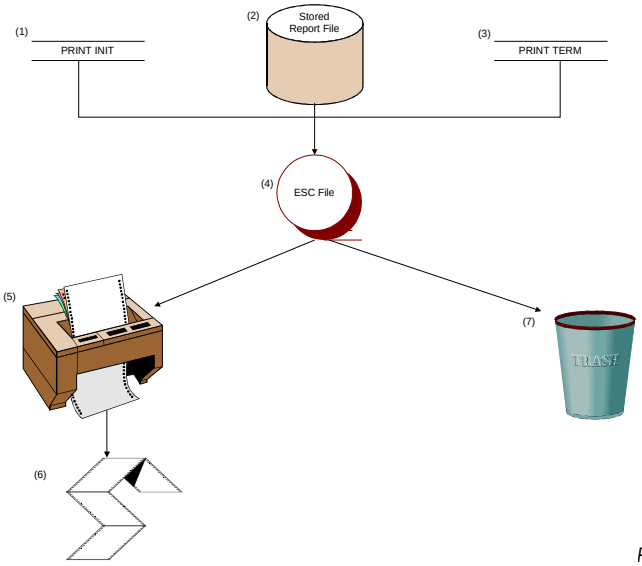
Caution: saving a DMS Window's contents by clicking File…⇨Save… will not create a useable report file because it will not contain the embedded FormFeed characters.

BUILDING A PRINT FILE UTILITY

After you have a systematic way of storing reports online, the next step is to develop a mechanism for sending those stored report files to the printer with the appropriate formatting instructions. Unfortunately, the forms you have so carefully developed cannot help you in this situation – they can only be used to control the printing of the contents of DMS windows. What you need instead is a utility program that provides an interface between the stored report output and the printer, just as the forms provide an interface between the contents of DMS windows and the printer.

The requirements for a print file utility are diagrammed in Figure 1.

The contents of the report file (2) are sandwiched between the PCL setup instructions derived primarily from the PRINT INIT section (1) and the PCL exit instructions from the PRINT TERM section (3) of the of the Printer Control Language screen to create the "printable" ESC file (4). The ESC file is transmitted to the printer using the *lp* command (UNIX) or the *type* command (Windows NT) (5). The printer recognizes the transmitted ESC file as a formatted document because of the PCL sandwich that surrounds the report file and prints off the report (6). The last responsibility of the utility is housekeeping (7) – deleting the ESC file using the *rm* command (UNIX) or the *del* command (Windows NT). The following macro shell takes the instructions from the UNIX form QLAND150 that we developed earlier and incorporates them into a print utility that will create formatted reports from report files in that same environment.



1 *Figure 1 Requirements for Print File Utility*

1

```
%macro rptprnt(rptloc);
%*******************************************;
%*   - create the printer set-up pcl         ;
%*       the &formdef macro variable is a hex ;
%*       string that contains pcl            ;
%*******************************************;
%let topmarg=8;
%let lftmarg=13;
%let rgtmarg=162;
%**** ESC & l 1 O -- landscape orientation ** ;
%** ~=Escape  ~ & l 1 O ******************** ;
%let formdef=1B266C314F;

%**** ESC ( 1 0 U -- PC-8 symbol selection ** ;
%**** ~=Escape         ~ ( 1 0 U ************ ;
%let formdef=&formdef.1B28313055;

%**** initial esc seq & fixed spacing: ~(s0p  ;
%**** number chars/inch:              16.67h  ;
%**** primary height:                   8.5v  ;
%**** primary style - upright (solid):    0s  ;
%**** primary font stroke weight - med:   0b  ;
%**** line printer type face:             0T  ;
%**** ~=Escape         ~ ( s 0 p 1 6 . 6 7 h  ;
%let formdef=&formdef.1B2873307031362E363768;
%****                  8 . 5 v 0 s 0 b 0 T    ;
%let formdef=&formdef.382E3576307330623054;

data _null_;
```

4 `file "&rptloc..esc";`

```
   infile "&rptloc..lst"
      end=eof
      length=inreclen;

   ****************************************;
   * - the set-up pcl is the first section of ;
   *   the "sandwich" file                   ;
```

```
   ****************************************;
   * replicate the ESC seqs from the form def ;
   outputst=trim("&formdef"x);
   put @ 1 outputst;

   * set left margin - 027 038 097 #...# 076 *;
   outputst=trim(byte(027)     ||
                 byte(038)     ||
                 byte(097)     ||
                 "&lftmarg"    ||
                 byte(076)
                );
   put @ 1 outputst;

   * set right margin - 027 038 097 #...# 077 ;
   outputst=trim(byte(027)     ||
                 byte(038)     ||
                 byte(097)     ||
                 "&rgtmarg"    ||
                 byte(077)
                );
   put @ 1 outputst;

   * set top margin - 027 038 108 #...# 069 **;
   outputst=trim(byte(027)     ||
                 byte(038)     ||
                 byte(108)     ||
                 "&topmarg"    ||
                 byte(069)
                );
   put @ 1 outputst;

   /* not needed after printer software upgrade
   * CR to register settings before 1st page *;
   put @ 1 byte(013);
   */
```

2

```
   ****************************************;
   *  "print" the report itself             ;
   ****************************************;
   do until(eof);
      input;
      put _infile_;
   end;
```

3

```
   ****************************************;
   * "print" the printer restore pcl as the   ;
   *    last section of the "sandwich" file   ;
   ****************************************;
   *** restore original settings - 027 E ;
   outputst=trim(byte(027)     ||
                 'E'
                );
   put @ 1 outputst;

   stop;

run;
```

```
%*****************************************;
%*  create filerefs with the UNIX command to  ;
%*  send the "sandwich" file to the printer   ;
%*  and to delete the "sandwich" file         ;
%*****************************************;
filename unixlp pipe "lp &rptloc..esc";
filename unixrm pipe "rm &rptloc..esc";
```

5

```
%*****************************************;
%*  issue the UNIX lp command to send the     ;
%*  "sandwich" file to the printer            ;
%*****************************************;
data _null_;
   file unixlp;
run;
```

7

```
%*****************************************;
%*  issue a UNIX rm command to delete the     ;
%*  "sandwich" file (this is done in a        ;
%*  separate data _null_ because the UNIXLP   ;
%*  subprocess does not execute when they are ;
%*  done within the same data step)           ;
%*****************************************;
data _null_;
   file unixrm;
run;


%*****************************************;
%*  clear the filerefs for the UNIX commands  ;
%*****************************************;
filename unixlp clear;
filename unixrm clear;

%mend rptprnt;
```

ENHANCEMENTS TO THE PRINT FILE UTILITY

Now that we have a program to control the formatting of printed output, we have the capability for building on enhancements that can add some of the capabilities we talked about earlier.

Recovering from a Paper Jam

To recover from a paper jam, we would want to print out only some of the pages from the report file – the ones the printer “ate.” To do this, we would still need all the same PCL, so the section of the program that manages the PCL would remain intact. Where we would need to make changes would be within the central DO UNTIL (EOF) loop that manages the report file. It would be fairly straightforward to insert a few lines of code there to conditionally PUT only the page (or range of pages) that we would specify with additional macro parameters.

Using Multiple Character Sets for the Same Report

We mentioned earlier the need for printing within a single report different special characters that are not part of the same character set. To accomplish this, we could use a more elaborate version of the same strategy we used to recover from a paper jam. Again, the PCL section would remain intact, and we would concentrate our efforts on the DO UNTIL (EOF) loop. There, we would need to be able to detect when the input buffer contains a character from a foreign character set. Once we encounter this situation, we would need to break the line into two segments immediately before the foreign character. Then, to the first segment, we would:

§ concatenate PCL to temporarily change to the foreign character set

§ concatenate the foreign character

§ concatenate PCL to change back to the report's native character set

§ concatenate the second segment.

On a smaller scale, this is the same type of sandwiching process that the print utility does to convert the report file into the ESC file.

CONCLUSION

After the number crunching is done, the biggest programming challenge in creating a report is to print it out according to user specifications. The SAS FORMS subsystem gives you a great deal of control over the output, but has several drawbacks. Using a form definition as the basis, it is possible to create a print file utility that not only overcomes those drawbacks, but also provides other functionality, like recovering from printer jams.

REFERENCES

SAS Institute, Inc. (1990), *SAS Language: Reference, Version 6, First Edition,* Cary, NC: SAS Institute, Inc., pp 811-821, 882.

Hewlett-Packard Company (1993), *The HP LaserJet 4L Printer,* Boise, ID: Hewlett-Packard Company.

ACKNOWLEDGMENTS

Gordon Davis and Donna Rogers (Intercardia, Inc.), and Paige Wright (Meridian Software, Inc.) for help with the UNIX/QMS printer features.

Eric Brinsfield and Peggy Hastie (Meridian Software, Inc.) for help with the Windows NT HP printer features.

Cathy Brinsfield (Meridian Software, Inc.) for editorial suggestions and for help with putting this paper in its final form.

TRADEMARKS