

Total Control of your Printed Output Without Desktop Publishing

Bill Adams, Adams Consulting Services

Overview

This paper will provide the reader with information on the philosophy behind and the techniques required to use advanced printer control in the generation of SAS® printed output. Examples will be given based on a H-P LaserJet III® printer generating output from a SAS OS/2 v6.08 session. This particular printer uses PCL5® as its printer control language, laser printers on other platforms may use a different control language but will have at least the capabilities demonstrated in this paper.

So what's wrong with PROC PRINT?

Nothing. PROC PRINT does an excellent job in producing legible output with a minimum of programming effort. As it has developed over the years, many options are available to tailor the report and the output produced is adequate for many applications. PROC REPORT has many more capabilities and is the standard method for report generation in many companies. The problem is that both are stock models with a list of options that were determined 'by committee', 'by consensus of opinion', 'by the Institute' but for sure 'by someone else'. It all boils down to the same thing as shopping for a new house. You have the tract houses made from stock plans with choices limited to carpet color and light fixtures. The next step is the semi-custom house with a much longer list of options but still a limited number of styles. At the high end is the custom home where you sit with an architect and a blank sketch pad and 'roll your own'.

As is true with many more sophisticated programming styles, additional time is usually required to use the techniques demonstrated in this paper. This may or may not be justified on a particular report. It is a decision based on time and money and needs to be made by those sponsoring the effort.

Why generate custom reports?

I think that everyone will concede that if PROC PRINT or PROC REPORT will do the particular job adequately, that's the way to go. In general, these procedures result in faster and cheaper output. So why generate custom reports? Simple, some managers want, and are willing to pay for, custom reports. They can specify exactly how they want them to look and SAS can generate that output. SAS might appear to be an expensive tool to do this type of reporting but it becomes quite cost effective when the

database is in SAS and the analysis for the report is being done in SAS. There can be many different types of custom reports just as there can be many types of custom houses. I'll limit this paper to custom listing reports used to support a large document produced by a state of the art word processing system like Microsoft Word for Windows® or WordPerfect for Windows®. Our goal is to make the listings produced by SAS look as good as the rest of the document. How many reports have you reviewed in which the supporting listings were basically photo-reduced copies of mainframe lineprinter dumps? Were you impressed? Did you feel confident that a high degree of professionalism was maintained throughout the data collection, analysis, and report generation phases of the project? No, you weren't impressed. A few hundred dollars were saved and now a multimillion dollar project is being questioned by the client or reviewing authority. This is not how to make money and have happy clients. Let's look at some techniques to minimize the cost and time required to do custom listing reports.

Write it once, use it many

It takes time to invent anything, that's where the cost and pain is. The trick is to figure out how to use that house plan, or program, or widget of code, over and over again so that the cost per use becomes quite low. One of the cravats is to not make your widgets too large or involved. Just as you might well be able to reuse a bay window design from a custom house, you probably won't be duplicating the entire floorplan for the first floor in another house. Instead of a bay window, let's design a set of tools to be used for the care and feeding of a laser printer using PCL5 as a control language — this covers most of the non-postscript high end printers used on micro and mini systems.

Look in the book

As I said before, a laser printer is capable of some pretty fantastic things. The printed page is basically a picture composed of black dots at a density of 300 dots per inch and you can control every one of those possible dot locations if you need to. Spend some time getting acquainted with the Technical Manual for your printer of choice. It is one of those thick books that may never have been opened since the printer was installed. This is where you'll find out what you need to tell the printer to make it go into bold typeface, or to change font, or to output 24-point headlines, or to draw a quarter inch square checkbox, or to print a custom logo from your client. Once you find the code what do you do with it? Find the code to cause the printer to print in bold type. It

When a computer wants a printer to print the letter 'a', it sends a unique series of 1's and 0's to the printer that represent the letter 'a'. This code is 8 bits long and is normally stated as a HEX or DECIMAL number. The normal code used outside of the IBM Mainframe world is ASCII and the character conversion tables in the back of any printer reference manual will list out the codes and the characters they represent. A total of 255 codes are possible with 8 bits, more than enough for all the letters, numbers and punctuation marks. The other codes are used to represent special symbols or in the case of codes up to '1F'X, to be used for device control. The expression '1F'X means the number 1F to the base 16 and is normally read as "ONE F HEX".

sure isn't 'make bold', it looks like Ec (s#B with the # being a number between -7 and 7. The associated text says that a stroke weight, that's really what we're talking about, of 0 is medium weight, -3 is light and 3 is bold. So it looks like the printer wants us to say Ec(s3B and it will start printing bold. OK, so now how do I talk to the printer?

Yo, Printer

"Yo" isn't quite the word the printer is listening for, it's really the Escape character. If you are not familiar with character code pages and hexadecimal notation, please read the sidebar on the previous page.

As a matter of convention, I will print hex strings bounded by single quotes and with a capital X suffix ['1B'X]. This is the standard SAS convention for writing hex character strings in code. OK, now the printer is listening so what do you tell it? Referring back to the code for bold stroke weight, we need to say '1B'X and '(s3B' to cause the printer to start printing bold with the current typeface and size. The '1B'X produces the escape character and the manual says '(s3B' will do the rest. One thing to remember, the printer has no way of knowing when to return to normal stroke weight unless you tell it. After you're done with the bold text, send the printer '1B'X '(s0B' and it will return to normal weight.

This simple example made two things very apparent. First, escape sequences are nasty looking things that make absolutely no sense when you read them. Second, it takes time to type them and a typo could make the printer do some very strange things. Another problem is that it is difficult to scan code and tell what the sequences are supposed to be doing without thumbing through the tech reference and re-inventing the wheel each time. The fix to the problem is very simple, global macro variables with names you can remember. A big advantage to using macro variables for the escape sequences is that the codes are all in one place. If you need to adapt to a new printer, a postscript or IBM 3800 for instance, the only modifications required will be to the

```
data _null_;
call symput('zE','1B'X);
run;
%let zWB = &zE.(s3B;
%let zWN = &zE.(s0B;
data _null_;
file 'xx' noprint;
put
'This is a test:'
"&zWB. Bold Text in"
"this section"
"&zWN. and now back to"
"normal weight again.";
run;
```

Sample 1

```
/* Run SAMPLE 1 to build macro variables */
data smpl2;
drop j i;
array xa(10) a1-a10;
array xf(10) $ 5 f1-f10;
do j = 1 to 25;
do i = 1 to 10;
xa(i) = 3*i;
if .48 < mod(xa(i),1) < .84 then xf(i) = "&zWB.";
else xf(i) = "&zWN.";
end;
output;
end;
run;

data _null_;
set smpl2;
file 'smpl2.lst' lrecl=150 noprint
n=ps col=c line=r;
array xa(10) a1-a10;
array xf(10) $ 5 f1-f10;
do i = 1 to 10;
c = (i*14)-13;
put @c xf(i) @c+5 xa(i) 6.3 @;
end;
put;
run;
```

Sample 2

table of escape sequences. This can be a real time saver when you port code to a new platform. Sample 1 contains a small code section that demonstrates how to set and reset the bold attribute from within a SAS data step. The thing that is significant here is not the printing of bold text, it is the fact that we can determine when bold text is printed by program control.

A more practical application would be to automatically bold a numeric value in a table listing depending on the results of a conditional test to another variable, perhaps a normal range limit. This type of report is only available with this technique. The other option would be to input the report into a word processing package, manually select the text, and then apply a bold attribute. This option would be very prone to human error. Sample 2 contains code to produce a page of output that demonstrates this application.

There are a few things to remember when embedding control sequences in the output stream. First, SAS sees the output going to a NOPRINT file so it will not put titles or page numbers on the pages automatically. Second, SAS will automatically put a page feed character at the top of each output page in a PRINT file, this doesn't happen automatically in this system and you should code a form feed 'OC'X in your header statement. Lastly, SAS thinks it is printing characters when it outputs a control sequence so it increments the column position counter [col= option on file statement] automatically. When the print string gets to the printer, the control sequence is stripped off and interpreted by the printer before the string enters the print line buffer. As a result the printed text appears to move left on the line by the number of characters in the control sequence. To allow SAS to print these 'additional characters' on its print line, increase the record length in the output file [lrecl= option on file statement]. If you bear in mind that SAS thinks it's printing all the extra characters but the printer knows it's not, you should be able to figure out much of what's happening. To generate the actual hardcopy output I usually do a system COPY command of the file to LPT1: or whatever port your printer is attached to.

Conclusion

The examples shown have been very simple. They demonstrate a concept but it is up to the user to develop this concept into a full kit of tools. The Hewlett Packard reference manual listed in the References contains a great deal of information on how a modern laser printer actually handles input data and how it treats the 'physical page'. This manual contains many examples and all of the printer commands but it is up to the user to build the code fragments.

A SAS programmer venturing into a printer reference manual is analogous to a builder walking into a large lumberyard. All of the pieces are there --- doors, roof trusses, floor joists, wall paneling. But it is up to the builder to select and use the pieces to make the homeowner's idea a reality.

Reference List

HP PCL 5 Printer Language Technical Reference Manual, Part No. 33459-90903:
Hewlett-Packard Company

SAS Language: Reference, Version 6, First Edition

Acknowledgments

SAS is a registered trademark of SAS Institute Inc.

LaserJet and PCL are registered trademarks of the Hewlett-Packard Company

Microsoft Word for Windows is a registered trademark of the Microsoft Corporation

WordPerfect for Windows is a registered trademark of the Novell Corporation

Author

Bill Adams

Adams Consulting Services

95 Highmeadows Cir.

Powell, OH 43065-9432

Phone 614-548-6785

Fax 614-548-7899

Internet badams@infinet.com