This paper describes a Macro system to aid users of the SAS system in report writing. Traditionally, customized reports are produced by writing PUT statements to place the heading text, body of the report, and footing text on the page. The disadvantages of this method are that it is tedious to program and difficult to change. The system described here changes the traditional coding technique with the use of a template and macro calls. The user enters the heading text, locations of variables, variable names and their formats, and the footing text into a text editor producing a template. This user defined template is processed by the macros to position the data on the page. The advantages of this system are 1) the user can see the format of the report before it is actually printed, 2) it is easier to layout the template than it is to write the PUT statements to achieve the same result, and 3) changes are more easily made.

INTRODUCTION

This macro system was originally designed to print patient listings for clinical drug trials. These listings are fancy PROC PRINTs. The system is equally suited to printing other types of tables. I use this system to produce reports for clinical drug trials. Reports of this type can be divided into three basic parts; headings, footings and body. Headings print at the top of each new page and consist of constant text and variables. Footings print at the bottom of each page and also consist of constant text and variables. The types of variables most often used in headings and footings are the values of BY variables that are constant for the page being printed. Page numbers or dates are often used in headings and footings. The body of a report usually consists of data for observations in a data set. In drug trials this would often be data on a patient's lab analysis or adverse events. The macro system described is designed to make it easier for programmers to manage these three components of a report.

SYSTEM DESCRIPTION

A template is the file where the layout of a report is stored. Templates are entered into an external file using a full screen text editor. Template files should be defined so that very long records can be entered. The system supports line sizes up to 199. Therefore template files should be defined with a record length of 200 or greater, because column one is reserved for template processor control. Variable length records may be used to conserve disk space.

Templates are processed by a macro (%TPRINT) to produce a global macro variable environment. When the variables are needed they are referenced by macros provided to generate code for printing the headings (%TH), footings (%TN) and body (%TB) of the report. These four macros plus one utility macro (%REVERSE) make up the template macro system. The macros use techniques that may be of general interest to users of the SAS system who are not interested in report writing. Macro %TPRINT uses CALL EXECUTE to execute %GLOBAL statements to create global macro variables from an executing data step. %TPRINT also constructs macro expressions and stores them in macro variables to be used in %IF statements in macros %TH and %TN. Macro %TH uses the PARMBUFF option so it can be defined to accept an unspecified number of parameters. Macro %REVERSE a utility macro is used in %TB. Its function is the same as the data step function reverse.

An example template shown in Exhibit 1 will be used for illustration. While this example is very simple it will illustrate the main features of the template system. These features are justified lines with variables in heading and footing text and value suppression for by groups in the body. Column 1 of the template is used to control the action of the template processor by designating the three parts of a template; Headings, Body, and Footings.

Heading lines are shown in lines 1 through 9 and start with an H in column 1. In the example, heading line 1 will be reproduced exactly as it appears. Lines treated in this manner become double quoted character constants in PUT statements produced by macro %TH. Note that this line is not enclosed in quotes, quotes in lines of this type will be reproduced. Lines 2, 3, and 4 in the example template illustrate the way automatic justification can be accomplished. By placing the justification indicator character before a quoted string, a justified line will be produced. Justification indicator characters are ?, \, and ! for centered, left and right justified respectively. The justification characters may be changed by the user in the call to macro %TPRINT if they are not on your terminal or are to be used as part of the heading text. Lines that are automatically justified specify the right hand side of an assignment statement. When justification is specified, macro %TH will create a character variable by defining its length. Then %TH will assign the variable's value as defined in the template line. Macro %TH names the variables with two underscores followed by the macro variable name to avoid possible problems with user defined variable names. _H2 for example. Lines 5 and 6 show how data step variables may be included in heading lines. The SAS system
character manipulating functions are especially useful for constructing lines of this type. Line 7 will be blank. Line 8 will be the column headings for the variables printed in the body of the report. Line 9 shows how the REPEAT function can be used to generate a dashed line. Macro variable &WIDTH, a %TH parameter, is the line size of the report. Subtract 1 from &WIDTH to make a dashed line of the correct length when using REPEAT.

Lines designating the body of the report are shown in lines 10, 11, and 12 of the example template. These lines define column locations, variable names, and formats for the body of the report. Line 10 begins with a C in column 1 and describes the column locations for printing variables. Line 11 begins with a V in column 1 and names the variables to be printed. Line 12 begins with an F in column 1 and names the formats for printing the variables named in line 11. Each variable name in line 11 must also have a format associated with it in line 12.

Footings lines are shown in Lines 13, 14, 15, and 16. These lines begin with an N in column 1 and specify the lines to be printed at the bottom of each page. This example uses only constants as in line 1 of the heading, however you may also use the justification and variable insertion techniques described for headings in the footing lines.

After the template has been constructed the following steps are used in the program:

1) %TPRINT is called before the data step that will print the report, making the template available to the other macros in the system;
2) %TH is called after the label statement named in the FILE statement’s HEADER= option;
3) %TN is called where the routine to print the footing lines would have been written;
4) %TB is called where put statements for printing the body of the report would have been placed in the data step.

The SAS system log from a program using the example template described in Exhibit 1 is shown in Exhibit 2.

Exhibit 3 is page one of the report produced by the example program shown in Exhibit 2.

Exhibit 4 is the code for the template system macros; %TPRINT, %TH, %TN, %TB, and %REVERSE. %TPUT, a macro not used in the example is also shown. Its function is similar to %TB. %TPUT is useful when the features of %TB are not required or do not produce the desired results. Parameters for %TPUT are similar to %TB, for example %TPUT(2,4,6) would print variables 2, 4, and 6 from a template.

For more information contact,
John Henry King
6613 Keystone Dr.
Raleigh, NC 27612
Phone (919) 846-8228

1SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA.
call execute('%!llobal n'liIeftlputtnO.3.JJII';
   call symputl 'n' liIeftl putt nO.3. I) ,triml line I I;
end~
otherwise put 'NOTE, Unknown m value input line bypassed';
end.
return;
eof.
if h_ifl=" then h_if I ::
   h_ifI = '%nrstr{'lTtrimlleftth_ifIIJlI'J';
call symput{ 'h_ifl ' .trimlh_ifl I I;
if h_ifZ=" then h_if2 =
   h_ifZ = 'Xnrstrl'lTtrimlleftlh_ifZJJlI' I';
call symputl 'h_if2'
   ~triml h_ifZ J I;
if h_if3=" then h_if3 
   h_if3 = 'Xnrstrt 'ITtrimlleftth_if3JIII, I';
call symput{ 'h_if3,trimlh_if3JJ;
call symput( 'hO',leftlputfhO,3. JIl;
call symput{ 'cO'
   ~leftlput(cO.3. ) h
   call symput{ 'vO',leftlputlcO.3. ))1;
call sYlllputl 'fO' .leftl putt cO,3. J) h
if n_ifl:" then n_if I = '0';
   n_Ul = 'Xnrstr( 'ITtriml leftl n_ifl I) II' J',
call symputt 'n_ifl' ,trimt n_ifl I);
   call symput( 'nO',leftlputlnO.3. Ill.
stop; run;
Page;
options noserror;
   %local i w parms;
   %let parms " %quotel I:bquotel &syspbuff Il%strl NULL);%if 1I.parms = NULL %then %let parms = Xstr{ I.NULL I;.
   %let i:1J
   %let w = %scant&parms.&il;
   %put NOTE, Parms=&parms;
   %whilel%'luotel &w J
   --let i I:evall 1I.i ... 1 I;
   %let w = %scanl&parms.&il;
lend.
   %mend Tput;
%macro Reverset arg );
   %local
   %do i = %lengthf &arg J I:to I %by -I;
   %let reverse
      = &reverse.%substrl larg .&i.l I;
   %end;
   &reverse
%mend Reverse;
%macro Tput / parmbuff;
   %local i w parms;
   %let parms " %quotel I:bquotel &syspbuff Il%strl NULL);%if 1I.parms = NULL %then %let parms = Xstr{ I.NULL I;
   %let i:1J
   %let w = Xcount(ampars,&il)
   %put NOTE, Parms=&parms;
   %let
   %do Zch=Zk2(quotel(&w)) = &zstr(NULL));
   %if Zstr = &zstr(NULL) %then %let reverse = Reverser.%substrlarg,di.
   %end;
   %put
%End Tput;
%macro Reverser(arg):