Paper 229-25

Variable and Format Consistency – A Macro Approach for Reading in Flat Files

Kevin P. Kowitz, Prime Therapeutics Inc, Eagan, MN

ABSTRACT

Many companies face obstacles in reading flat files due to discrepancies in naming and formatting conventions. In this scenario, changing a record length and/or a variable length or format requires global changes to all programs. This task is difficult in a non-production environment where every program may differ in variable names and formats. To simplify this process at Prime Therapeutics, Inc; a SAS® Macro was implemented for reading in flat file data. This Poster describes the SAS table and Macro used for this process.

INTRODUCTION

While analyzing the effect of Y2K on SAS programs, it was discovered that there was little consistency in how individuals named and formatted data at Prime Therapeutics. In order to solve the problem, a SAS lookup table was created to standardize variable names and formats. The Macro was created to read in the table and put out a data step View.

TABLE

To develop a consistency in variable names and formats the following dataset was created. This dataset was produced after developing a standard for each variable name and format.

CONTENTS PROCEDURE

Data Set Name:	MAC.VARIABLE	Observations:	99
Member Type:	DATA	Variables:	3
Engine:	V612	Indexes:	2
Created:	15:36 Fri, Oct 1, 1999	Observation Length:	24
Last Modified:	15:36 Fri, Oct 1, 1999	Deleted Observations:	0
Protection:		Compressed:	NO
Data Set Type:		Sorted:	NO
Label:	DEFAULT VARIABLES		

-----Engine/Host Dependent Information-----

Data Set Page Size:	8192
Number of Data Set Pages:	1
File Format:	607
First Data Page:	1
Max Obs per Page:	338
Obs in First Data Page:	99
Index File Page Size:	8192
Number of Index File Pages:	3

-----Alphabetic List of Variables and Attributes-----

#	Variable	Type	Len	Pos
1	POSITION	Char	8	0
3	VARFMT	Char	8	16
2	VARNAME	Char	8	8

Example of Data from the above dataset:

POSITION	VARNAME	VARFMT
@389	FIELD1	$CHAR8.
@194	FIELD2	$CHAR15.
@105	FIELD3	YYMMDD6.

MACRO CODE

A macro was created to use the above table in the following ways.

1) Determine what variables were being requested.
2) Determine if the variable had an approved name.
3) Abort the program if an approved variable was not selected.
4) Create a View of the Flat file to be read in.

The following is the code that was developed.

```
%macro flat(dbname,infile,keeper);
%*******************************************************************%;
%*  MACRO: FLAT     CREATED BY: Kevin Kowitz                       %* ;
%*                                                                 %*;
%*  USAGE: 1) %flat(dbname,infile,keeper)                          %*;
%*                                                                 %*;
%*  DESCRIPTION:                                                   %*;
%*   This macro creates a view of a flat file for use within       %*;
%*   a SAS program.                                                %*;
%*                                                                 %*;
%*  INPUT:                                                         %*;
%*   DBNAME - This is the libref for the view created by           %*;
%*          this macro.  It can be accessed by another             %*;
%*          program if it is saved as a permanent view.            %*;
%*                                                                 %*;
%*   INFILE - This is the name of the fileref that the view is     %*;
%*         created for.  This fileref is assigned in the main      %*;
%*         program. (i.e. filename infile '/path/filename')        %*;
%*                                                                 %*;
%*   KEEPER - This is the list of variables that are to be         %*;
%*          read in.  The list should be space or carriage         %*;
%*          return delineated                                      %*;
%*          DO NOT QUOTE THE STRING.                               %*;
%*                                                                 %*;
%*   e.g. %flat(dbname=work,infile=filein,keeper=field1            %*;
%*                                          field2   field3)       %*;
%*                                                                 %*;
%*******************************************************************%*;
OPTIONS NOSYMBOLGEN NOMACROGEN NOMPRINT
NOMLOGIC;

LIBNAME MACRO1 'Directory with LookupTable mac.variable in it';
%*Initialize variable to abend job if unapproved variables used *;
%let jobend=0;

%*******************************************************************%*;
%* The next code snippet takes multiple spaces between             %*;
%* variables out of &keeper.  This is a  copy of the autocall      %*;
%* macro %qcmpres.  The macro %qcmpres needed to be                %*;
%* included in this macro to remove a nested macros error          %*;
%*******************************************************************%*;

%let keep=%qleft(%qtrim(&keeper));
%let i=%index(&keep,%str( ));
%do %while(&i ne 0);
%let keep=%qsubstr(&keep,1,&i)%qleft(%qsubstr(&keep,%eval(&i+1)));
%let i=%index(&keep,%str( ));
%end;
```

```
%*********************************************************************%*;
%* The next code snippet counts the number of words in the           %*;
%* variable &keep1 The 7 lines of code starting at                   %*;
%* %let varnum=1, is a modification of a word counting               %*;
%*  macro found in the publication                                   %*;
%* "SAS® Guide to Macro Processing                                   %*;
%*  Version 6, Second Edition                                        %*;
%*   Page 256.                                                       %*;
%* It was necessary to do this all with Macros due to                %*;
%* the 200 Character size limitation in datastep processing          %*;
%*********************************************************************%*;

%let varnum=1;
%let word=%qscan(&keep,&varnum,%str( ));
%do %while(&word ne);
    %let varnum=%eval(&varnum+1);
    %let word=%qscan(&keep,&varnum,%str( ));
%end;
%let varnum=%eval(&varnum-1);

%*********************************************************************%*;
%* Create a table with an observation for each field                 %*;
%* requested.   Use macro functions to scan in the variable          %*;
%* names and output them to the table. This table is then            %*;
%* merged with the lookup table so the position and format           %*;
%* of the variables can be determined.                               %*;
%*********************************************************************%*;
data tempvar (keep=varname);
   length varname $10.;
   %do i=1 %to &varnum;
     varname="%upcase(%scan(&keep,&i))";
     output;
   %end;
run;

**sort by varname for merging with approved variable names **;
proc sort data=tempvar;
  by varname;

%*********************************************************************%*;
%* Merge with approved names from lookup table                       %*;
%*********************************************************************%*;
%* macrovar = concatenation of position,name & format *              %*;
%*********************************************************************%*;
%* If a variable was selected in the program that is not             %*;
%* an approved variable name reset jobend to stop the job            %*;
%*********************************************************************%*;
data varsubmt;
merge tempvar (in=a) macro1.variable (in=b);
by varname;
if a;
 format macrovar $char25.;
   if position ne ' ' then do;
   macrovar=left(trim(position))||' '||left(trim(varname))||' '||left(trim(varfmt));
   end;
   else do;
     call symput('jobend',1);
     call symput('endvar',left(trim(varname)));
   end;
run;

%*********************************************************************%*;
%* If &jobend is set to 1 then abend the job and printout the        %*;
%* approved list of variable names.                                  %*;
%*********************************************************************%*;
%if &jobend=1 %then %do;
  proc sort data=macro1.variable out=variable;
  by position;



  data _null_;
  set variable;
    if _n_=1 then put
       "The variable name &endvar is not an approved name."/
       "Please select your variable names from the following list."//;
    put @10 position $char4.
        @20 varname  $char10.
        @30 varfmt   $char8.;
  run;

  data _null_;
  abort;
  run;
%end;

%*********************************************************************%*;
%* Transpose the dataset varsubmt so kept variables may              %*;
%* be read into macro variables in the next step                     %*;
%*********************************************************************%*;
proc transpose data=varsubmt out=vars;
var macrovar;

%*********************************************************************%*;
%* Write the Kept Variables in COL1-COL(n) into the macro            %*;
%* variables VAR1-VAR(n).  The macro variables will be               %*;
%* used  in the input section of the next datastep.                  %*;
%*********************************************************************%*;
data _null_;
  set vars;
    %do i= 1 %to &varnum;
     call symput("var&i",COL&I);
    %end;
run;
                    (The above code starting from the transpose
                     step could be replaced with the following
                     proc sql noprint;
                     select macrovar into :var1-:var&varnum
                     from varsubmt;
                     quit;)
%*****************************************%*;
%* Create the View &dbname..outfile.    %*;
%*****************************************%*;

data &dbname..outfile / view=&dbname..outfile;
 infile &infile lrecl=570 recfm=f end=eof;
   %do i = 1 %to &varnum;
     %if &i = 1 %then input &&var&i;
     %else &&var&i;
   %end;
   ;        /** The ";" is needed to complete the input statement **/
run;
option symbolgen;
%mend flat;
```

EXAMPLE

The following is an example of reading in the variables FIELD1 and FIELD2 sorting the output by FIELD1.

```
Libname save '/home/directory';
filename example '/example/dataset';

%claims(save,example, field1 field2);

proc sort data=save.outfile out=example1;
by field1;
where field1 = 'Whatever you want to limit the file by';
run;
```

ADVANTAGES OF USING THE MACRO

1) Y2K conversions when smoothly because all changes to formats and variables were done in the lookup table and did not require editing every program.
2) Every program at Prime Therapeutics Inc, now uses the same variable names and formats when accessing flat files that have been set up to use this process.
3) Changing the Record length of the flat file only required changes to the macro and lookup table.
4) Doing statistics on variable usage is easier when there is conformity in naming conventions

DISADVANTAGES OF USING THE MACRO

1) There was approximately a 10% increase in CPU time for reading in some tables. This was due to an inability to read a record across iterations of the DATA Step. (i.e., use of @@) Depending on the size of your flat files this may prove a limiting factor.
2) Jobs are forced to abend if approved variable names are not used.
3) All programs and programmers need to be required to use the macro in order for the advantages of utilizing this method to be realized.
4) Any sort step using the view would need to output to a different table name.

CONCLUSION

The final result of requiring the use of the above macro has been consistency in variable names and formats. It has also proven effective in minimizing Y2K file changes from 2 digit to 4 digit years. Although an increase in CPU time was realized, the use of WHERE clause processing helped improve performance in this area. I recommend using a method similar to this if variable name consistency is a problem at your company.

REFERENCES

SAS Institute Inc., *SAS® Guide to Macro Processing, Version 6, Second Edition,* Cary, NC: SAS Institute Inc., 1990, 256pp.

.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Kevin Kowitz
Prime Therapeutics Inc.
P.O. BOX 64812
St. PAUL, MN 55164

Work Phone: 651.286.4031
Fax: 651.286.4400
Email: kkowitz@primetherapeutics.com