Paper 76-25

SAS' Best Kept Secret: Macro Windows® for Applications Development

Kay Alden, Consultant, Columbus, IN

ABSTRACT

SAS Macro Windows provide a quick solution for developing applications on the run. Time constraints or lack of SAS AF® expertise on-site may necessitate the use of Macro Windows.

INTRODUCTION

Macro Windows are easy to construct and have various options to allow the developer to control the size, content and even color of the window. Macro programs are used to drive the application. A thorough working knowledge of SAS Macro language is helpful.

Application

The application presented here allows the user to perform statistical analysis without the intervention of a programmer and/or statistician. The project design was coordinated with the project statistician and the user. A series of macro programs define the windows, allow access to project data files, perform edit checks and run the requested statistical analysis. The controlling program is the macro guide. It contains all of the window definitions and invokes a macro program (demog) which displays the first window. This program checks for a valid filename and performs edit checks. Other windows are displayed and macros invoked depending on user input and/or the successful completion of edit checks. If a filename is entered incorrectly, a window opens telling the user that it is an invalid filename and to try again. Most of the macro code used in this application checks values of macro variables, assigns values to macro variables and opens new windows. Data steps and proc steps are used to generate the requested reports. The reports are viewed in the output window. Remember that data step processing can be accomplished inside a macro.

%WINDOW

The %window statement is used to create a window. Fields can be created to accept user input or to display information back to the user. The following code creates the first window used in a specific application.

```
%window main columns=120   rows=35 irow=1 icolumn=1
   #2 @5 'Welcome to the Beluga Analysis System'
   #4 @ 5 'Please enter the Project Name and Number for
        this analysis:'
   #6 @ 5 'Project: '@14 project 75 attr=underline
   #8 @5 'Project Number:' @24 projnum 15
       attr=underline
   #10 @ 5 'Please Indicate if data is for Children or
         Adults.'
   #12 @ 5 'Place an X in the appropriate choice:'
   #14 @15 'Adults:' @25 adult 1 attr=underline
   #16 @15 'Children:' @25 child 1 attr=underline
   #18 @ 5 'Please enter filename for Demographic data:'
   #20 @10 demo 50 attr=underline
   #25 @ 5 'Press Enter to Continue' ;
```

It's best to create a unique name for each window in the application. The name of the window is specified directly after the %window keyword. The name of the window being defined above is main. Specific attributes allow for customization.

Sizing the window

The rows= and columns= attributes control the size of the window. In the sample code, the window is defined as 120 columns wide and 35 rows long. The irow= and icolumn= attributes specify a starting place for the window on the monitor screen. In this example row 1 and column 1 were specified as starting points on the monitor window. The sizing of the window is the trickiest part of using Macro Windows. The appearance of a window depends on the monitor, resolution, and font used to display the window. When these items differ between the developer and the user, the sizing of the macro window may need to be adjusted.

Other window options

There is also a color= option to change the background color of the screen. This is useful in drawing the users' attention to a particular window, such as one that reports errors. There are other options available such as menu= for defining a pmenu for the window. A keys= option can also be specified to provide function key definitions for the window. This application did not make use of either of these options.

Placement of text and fields

The first line of the window, identified with a line number specification (#2), is fixed text. Note that the line number specification is similar to that used on an input statement in the data step. Column indicators are also used to specify placement of text on a given line. It's important to remember to enclose fixed text in quotes.
User input is requested on the third line of the window which is defined as line 6(#6). The name of the project is requested and stored in a macro variable called project. The value of the variable starts in column 14(@14) and is 75 columns long. The 75 immediately following the variable name defines the length of the display field. Even though this is a macro variable, only the name of the macro variable is used, not the macro variable reference such as &macrovar. To prevent possible overlapping of multiple fields in a window, SAS documentation recommends defining a length for each display field.

Field Attributes

The attr= option allows you to specify attributes for each field. In the example, attr=underline is specified. The user can see how long the field is and exactly where to place the information requested. Other field attributes are blink, highlight and rev_video. Several attributes can be specified after the attr= as long as they are enclosed in parentheses and separated by commas. Again the appearance of these attributes is dependent on the type

of monitor being used to display the window Fields can also be protected and/or required by specifying the protect= and required= attributes. These attributes are particularly useful in windows since they provide feedback to the user. In this particular application, a macro program checks the validity of the filename and the integrity of the data. When invalid data are encountered in the requested file, a window is displayed providing the user with the name of the file and instructions to check the output window for a listing of the errors. It is permissible to run the analysis with data errors, so the protect= attribute is specified in this secondary window to prevent the user from changing the value of the macro variable containing the name of the file.
As with any SAS statement the %window statement ends with a semi-colon.

%Display

The %display statement is used to open a window. To view the window defined above simply submit a %display statement specifying the name of the window, such as %display main;. If user input is required, the user must provide the requested information and hit enter to continue. For a window that displays information only but does not have a field requiring user input, the user must still hit enter to proceed.

How SAS processes windows

When a macro containing a %window statement is compiled, SAS stores the statement as text. At execution time, the window definition is stored in a utility file in the WORK library. Every time a %window statement is executed, the window file is recreated. SAS documentation recommends placing window definitions in a controlling program, which is executed once. Macro variables are stored in the current referencing environment.

Two of the macros from this application are included in the appendix.

Training

Most of the training included showing the user how to start SAS Display Manager®. Function key definitions were provided for each of the Display Manager windows. Since the user had been consulted during the design of the windows, most of the windows were self-explanatory. One to two hours were dedicated to training. Support was minimal after that.

CONCLUSION

This particular application was developed using macro windows due to time constraints. An AF applications developer was not available. Due to a short project deadline and budget constraints, AF training was not feasible. Macro windows provided a great alternative for someone well versed in macro language and BASE SAS software. The application was developed within a one to two week timeframe, including testing and training.

REFERENCES

SAS Institute Inc. (1990), *SAS Guide to Macro Processing, Version 6, Second Edition*, Cary, NC, SAS Institute Inc.

ACKNOWLEDGMENTS

Special thanks go to my husband for all of his support and encouragement, to my children for putting up with me while I was working on this. Special thanks to Ben Cochran, Jon Kosanke and Lavon Missell for their suggestions, comments and editing help.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. You can contact the author at:

Kay Alden
Consultant
6111 Sloan Valley Dr.
Columbus, IN 47203
Phone: (812) 375-9557
Email: kalden@kiva.net

APPENDIX

The following macro contains some of the window definitions for this application. Due to space constraints, not all of the windows for this application have been included.

```
%macro guide;
options mprint mlogic;
goptions reset=all;
title;
footnote;
%window main columns=120   rows=35 irow=1 icolumn=1
   #2 @5 'Welcome to the Beluga Analysis System'
   #4 @ 5 'Please enter the Project Name and Number for
        this analysis:'
   #6 @ 5 'Project: '@14 project 75 attr=underline
   #8 @5 'Project Number:' @24 projnum 15
        attr=underline
   #10 @ 5 'Please Indicate if data is for Children or
         Adults.'
   #12 @ 5 'Place an X in the appropriate choice:'
   #14 @15 'Adults:' @25 adult 1 attr=underline
   #16 @15 'Children:' @25 child 1 attr=underline
   #18 @ 5 'Please enter filename for Demographic data:'
   #20 @10 demo 50 attr=underline
   #25 @ 5 'Press Enter to Continue' ;

 %window fixdata columns=75 rows=25 irow=1 icolumn=1
   #5 @ 5 'Data problems in:' @25 type 16 attr=underline
        protect=yes
    #9 @ 5 'Do you want to stop and fix data'
    #11 @ 5 'Or Do you wish to continue? (Enter s to stop
          or c to continue)'
    #13 @10 yesno 1 attr=underline;

 %window datok columns=50 rows=25 color=blue irow=1
       icolumn=1
    #5 @ 5 'Please check output window for reports'
    #10 @ 5 'Press Enter when ready to continue’;

 %window outliers columns=65 rows=25 color=blue
       irow=1 icolumn=1
    #5 @ 5 'Please check output window for reports'
    #7 @ 5 'Are data values out of range?(Y/N)' @40
        outyn 1 attr=underline
    #11 @5 'Press enter to continue.';
 %window exit columns=80 rows=25 irow=1 icolumn=1
    #5 @5 'Do you wish to run another analysis on a
          different study?'
    #7 @15 'Y/N' @20 yesno 1 attr=underline

    #11 @5 'Note: If you respond no, you will end this
           session'
    #13 @5 'Press Enter to Continue';

%window tryagain columns=95 rows=25 irow=1
      icolumn=1
    #5 @5 "Sorry, the File:"
    #7 @10 wrong 50 attr=(underline,rev_video)
         protect=yes
        @62 "doesn't exist."
    #9 @15 'Please try again'
    #11 @15 filenm 50 attr=underline
    #15 @5 'Press Enter to Continue';

 %window baddata columns=100 rows=25
    #5 @5 'Bad Data encountered in ' @35 filenm 50
         attr=underline protect=yes
    #7 @5 'Please check Log Window for error
         messages'
    #13 @5 'Please check Output Window for Listing of
           the Data'
    #16 @5 'Press Enter to End Session';

 %let wrong=;
 %let filenm=;

%demog     ***call to first macro in the application which
              displays the first window(main);

%mend guide;
```

The demog macro referenced above is listed next. It defines macro variables and formats for displaying values in certain fields. If the correct filename is specified, the file is read and processed. Other windows are opened depending on the success or failure of certain conditions. This is the only macro included from this application due to space constraints. Please contact the author for further information.

```
%macro demog;

%if %quote(&filenm)^= %then %let demo=&filenm;
  %else %do;
      %let wrong=;
      %let adult=;
      %let child=;
      %let demo=;
      %let project=;
      %let projnum=;
      %let norun=;
      %display main;
  %end;

  %if %quote(&demo) = %then %do;
    data _null_;
      call symput('norun','NO');
    run;
  %end;

 %if &norun= %then %do;

options nocenter;

data demo bad;
  infile "&demo" missover end=eof;

%if &adult ^= %then %do;          %**** read data
               values  depending on child vs adult;
  input eval age sex educ tried;
  format sex sexfmt. educ educfmt. tried trdfmt.
      age adultfmt.;
%end;
%else %if &child ^= %then %do;
  input eval educ age sex tried;
  format sex sexfmt. educ chedcfmt. age chldfmt. Tried
       chtrdfmt.;
%end;
  if _error_=1 then do;
      output bad;
      cnt + 1;
  end;
  output demo;
  if eof then call symput('bad',left(put(cnt,3.)));
run;

%if &bad^=0 %then %do;
      proc print data=bad;
      title 'Demographic data with possible errors';
      %display baddata;
%end;
%else %do;
%put syserr=&syserr;
%if &syserr^=0 %then %do;
    %let wrong=&demo;
    %let filenm=;
    %let norun=;
    %display tryagain;
    %demog
 %end;
%else %do;
     Proc freq data=demo ;
        tables age sex educ tried;
     data ck1 ;
          set demo end=eof;
          if age=. | sex=. | educ=. | tried=. then do;
                output;
                nobs + 1;
          end;
          if eof then call symput('nobs',left(put(nobs,3.)));
     proc print data=ck1;
     title 'Data with missing values for age, sex,
education or tried';
     run;

    *******check for duplicate eval numbers;

     proc sort data=demo;
       by eval;
     data dups;
        set demo end=eof;
        by eval;
        if ^(first.eval & last.eval) then do;
              output;
              nobs1 + 1;
         end;
         if eof then call
                 symput('nobs1',left(put(nobs1,3.)));
     proc print data=dups;
```

```
          var eval age sex educ tried;
          title 'Duplicate entries for this Evaluator';
        run;

        %let yesno=;
        %let noyes=;
        %let type=Demographic data;
        %let resp=;
        %let numatt=;
        %let filenm=;
        %let outyn=;

        %do i=1 %to 12;
             %let att&i=;
             %let label&i=;
             %let max&i=;
        %end;
    %if &nobs^=0 or &nobs1^=0 %then %do;
             %display fixdata;
             %if %upcase(&yesno)=C %then %resp;
        %end;
        %else %if &nobs=0 and &nobs1=0 %then %do;
            %display outliers;
            %if %upcase(&outyn)=Y %then %display fixdata;
            %if %upcase(&outyn)^=Y or %upcase(&yesno)=C
                    %then %resp;
        %end;
     %end;
  %end;
  %end;
%mend demog;
```