Creating A User Interface Using the SAS® Macro Window

George P Sharrard Ph.D.
GPS Corp

Abstract
The best part of writing reusable code is when you effect a hand-off to the end user. There is no good reason to have a highly skilled programmer running a program that any junior marketing analyst could run. Using a series of %LETs for all the parameters in the program makes it ‘marketing’ proof. Except, of course, when they can’t re-run the code without trying to make it better. So, you hide the code in a macro and just give them the syntax for the macro call. But, this too, will fail the ‘finance department’ stress test when parameters are entered incorrectly - dooming the user to long waits for bogus output.

What you need is a user interface, a parameter entry screen, which will validate the parameters entered before any programs are run. Do the input files exist? Do the requested variables exist on the datasets? Add in an option for creating a random sample or check-off boxes for various types of outputs. Base SAS includes all the tools needed to make this happen. Granted, it’s not beautiful to look at but it gets the job done.

Step 1. The Simple Program

```
filename in1 "c:\xyz_corp\data\outside_sales.dat" ;
libname saveme "c:\xyz_corp\data" ;

data saveme.oss_nyc_200506 ;
  infile in1 lrecl=80 ;
  input id 1 - 3
  gender $ 4
  age 5 - 6
  last_order_date yymmd8.
  first_order_date yymmd8.
  last_order_value 7.2
  sum_value_all_orders 6.2
  reference_codes $ 36 - 64
;#
format last_order_date first_order_date date9. ;

mail_cd=substr(reference_codes,1,4) ;
prod_ref=substr(reference_codes,5,6) ;
lem_cd=substr(reference_codes,7,1) ;
run ;

Proc freq data=saveme.oss_nyc_200506  ;
  tables mail_cd prod_ref lem_cd ;
run ;

Proc means data=saveme.oss_nyc_200506  min max mean ;
  var age last_order_value sum_value_all_orders ;
run ;
```
The things that will change are the filename and the meaning of reference_codes.

Each source of sales data will encode 1 – 3 codes in the reference_codes field. The order and size of these fields is variable. Two external sales offices may send mail_cd, but, one may send a 4 position mail_cd and the other may send a 3 position mail_cd. The 4 position mail_cd may appear at the end of the reference_codes field while the 3 position mail_cd may be the first. The tables statement on the Proc Freq will have to be modified to reflect the variable names from the decoded reference_codes field.

Because XYZ Corp. requires monthly reporting of sales data – multiple external sales data files will appear in the first days of the month, so we will want to make it easy to separately process multiple files.

What you have here is a process with a built in failure feature. You are depending on a user who only runs this program once a month to open your source code make modifications and run over multiple iterations of input files. Forgetting to change the filename or adjust the ‘meaning’ of the reference_codes field or the output dataset parameters dooms your program to failure!

**Step 2. The Macro Solution**

```sas
%macro readin(fi,nm1,st1,for1,nm2,st2,for2,nm3,st3,for3,loc,dt) ;
filename in1 "c:\xyz_corp\data\&fi" ;
data oss_&loc._&dt ;
  infile in1 lrecl=80 ;
  input id   1 - 3 gender $ 4 age   5 - 6
  last_order_date   yymmdd8. first_order_date   yymmdd8.
  last_order_value   7.2
  sum_value_all_orders   6.2
  reference_codes $ 36 - 64
;
format last_order_date first_order_date date9. ;
&nm1=substr(reference_codes,&st1,&for1) ;
&nm2=substr(reference_codes,&st2,&for2) ;
&nm3=substr(reference_codes,&st3,&for3) ;
run ;
%mend readin ;
```

We evoke macro `readin` using the following code.

```sas
options mautosource NOMprint NOMlogic NOsymbolgen NOsource NOsource2
       nofmtterr errors=1 nodate
sasautos=('C:\xyz_corp\macros'
          sasautos) ;
```
This is a major improvement. Now it is much simpler for the end user make mistakes! It’s obvious what will happen. The macro call:

\%readin(outside_sales.dat, mail_cd, 1, 4, prod_ref, 5, 6, lem_cd, 7, 1, NYC, 200506) ;

will be copied as many times as there are input files – edited – and the whole thing submitted. So, the end user will make something like this:

\%readin(city_sales.dat, off_cd, 1, 3, lem_cd, 4, product, 5, 5, NYC, 200506) ;
\%readin(sprin_hart_sal.dat, sku, 1, 4, mail_cd, 5, 6, comm_cd, 11, 8, CT, 200506) ;
\%readin(denver.dat, mail_cd, 1, 4, prod_ref, 5, 6, lem_cd, 11, 1, CO, 200506) ;
\%readin(balt_dc.dat, sale_ref, 1, 8, chan_cd, 9, 2, sku, 10, 8, Capital, 200506) ;
\%readin(suff.dat, mail_cd, 1, 4, sku, 5, 8, lem_cd, 13, 1, LI, 200506) ;
\%readin(boston.dat, product, 1, 7, effort, 8, 4, ref_cd, 12, 5, Boston, 200506) ;

They may or may not save this ‘program’ for later reference when the saved files are found to be incorrect. This is just too hard to proof read.

A slight improvement would be to give the user something that looks like this:

\%readin(fi=city_sales.dat,
    nm1=mail_cd,
    st1=1,
    for1=4,
    nm2=prod_ref,
    st2=5,
    for2=6,
    nm3=lem_cd,
    st3=7,
    for3=1,
    loc=NYC,
    dt=200506
    ) ;

But, this is only a slight improvement. The evil copy/paste demon is bound to make an appearance. So, what are we to do?
Step 3. The Simple Macro Window

What we have here is a simple parameter entry screen. The SAS code to make this screen follows:

```sas
options mautosource NOMprint NOMlogic NOsymbolgen NOsource NOsource2
nofmterr errors=1 nodate
sasautos=('C:\xyz_corp\macros'
          sasautos) ;
%macro startme ;
%global fi nm1 nm2 nm3 st1 st2 st3 for1 for2 for3 loc dt ;
%window first
#3 @45 'Outside Sales Data' color=blue
#6 @5 'Input Dataset' color=blue //
@5 'Outside Sales File Name:' color=black +2 fi 83 color=green required=yes
attr=underline //
@5 'Name of Field 1:' color=black +2 nm1 25 color=green required=no attr=underline
+2 'Starting Position:' color=black +2 st1 2 color=green required=no attr=underline
+2 'For How Many:' color=black +2 for1 2 color=green required=no attr=underline //
@5 'Name of Field 2:' color=black +2 nm2 25 color=green required=no attr=underline
+2 'Starting Position:' color=black +2 st2 2 color=green required=no attr=underline
+2 'For How Many:' color=black +2 for2 2 color=green required=no attr=underline //
@5 'Name of Field 3:' color=black +2 nm3 25 color=green required=no attr=underline
+2 'Starting Position:' color=black +2 st3 2 color=green required=no attr=underline
+2 'For How Many:' color=black +2 for3 2 color=green required=no attr=underline //
@5 'Location:' color=black +2 loc 12 color=green required=yes attr=underline
+2 'File Date:' color=black dt 6 color=green required=yes attr=underline
+2 '(in the form 200501)'
;
%display first ;
%readin2 ;
```
What we have is a macro called ‘STARTME’ which contains the code for window ‘FIRST’. We get to see window FIRST with the command %display first. Screen 1 at the top of this section shows the resolution of window FIRST. Let’s look at the code and what it does.

#3 @45 'Outside Sales Data' color=blue ////

Go to row 3 of the screen (#3), move over to column 45 (@45), write “Outside Sales Data” and color the text blue. Next...

#6 @5 'Input Dataset' color=blue //
@5 'Outside Sales FileName:' color=black +2 fi 83 color=green required=yes attr=underline //

Now drop down to screen row 6 (#6), move over to column 5 (@5), write ‘Input Dataset’, color the text blue and drop down 2 rows (//). Move over to column 5 write ‘Outside Sales Filename:’ using black text. Move over two columns from wherever you are (+2) create a macro variable named ‘fi’ allowing 83 space for user entry – color this entry field green – this is a required field – and show the field on the screen as 83 green underlines.

That’s it! You repeat this for each of our input fields. But, we can do better.

The Final Macros

What follows are the final version of the parameter window, the %readin3 macro and a new macro %decide. Each numbered boxed section will be described in detail at the end. As a final usability measure we will create a desktop icon to start out application.

%macro readin3;
filename in1 "&fi";
libname saveme "c:\xyz_corp\data";
data saveme.oss_&loc.&dt ;
infile in1 lrecl=80 ;
  input id 1 - 3
  gender $ 4
  age 5 - 6
  last_order_date yymmdd8.
  first_order_date yymmdd8.
  last_order_value 7.2
  sum_value_all_orders 6.2
  reference_codes $ 36 - 64
;
  format last_order_date first_order_date date9. ;
%if &nml ne %then %do ;
# 1
&nm1=substr(reference_codes,&st1,&for1) ;
%end ;
%if &nm2 ne %then %do ;
    &nm2=substr(reference_codes,&st2,&for2) ;
%end ;
%if &nm3 ne %then %do ;
    &nm3=substr(reference_codes,&st3,&for3) ;
%end ;
run ;

Proc freq data=saveme.oss_&loc.&dt ;
    tables &nms ;
run ;

Proc means data=saveme.oss_&loc.&dt min max mean ;
    var age last_order_value sum_value_all_orders ;
run ;

%startme ;
%mend readin3 ;

options mautosource NOmprint NOmlogic NOsymbolgen Nosource
    NOsource2 nofmterr errors=1 nodate
    sasautos=('C:\xyz_corp\macros'
                sasautos)
    ;

dm 'AWSMAXIMIZE' ;
dm 'wdockview' ;
dm 'log off' ;
dm 'WSTATUSLN off' ;
dm 'WWINDOWBAR off' ;
dm 'zoom' ;
data _null_ ;
run ; quit ;

%let fi=c:\xyz_corp\data;
%let nm1=; %let nm2=; %let nm3=;
%let st1=; %let st2=; %let st3=;
%let for1=; %let for2=; %let for3=;
%let loc=; %let dt=;
%let oops=; %let oops2=;
%let nms=; %let alldone=;

%macro startme ;
%global fi nm1 nm2 nm3 st1 st2 st3 for1 for2 for3 loc dt nms ;
%window first
"&oops" color=red / "&oops2" color=red
Programming & Manipulation

Outside Sales Data

Input Dataset

Outside Sales FileName:

Name of Field 1:
Starting Position:
For How Many:

Name of Field 2:
Starting Position:
For How Many:

Name of Field 3:
Starting Position:
For How Many:

Location:
File Date:
(in the form 200501)

Exit w/o Running:
Clear Messages:

**************** Hit ENTER to run ******************

%display first ;
%if &clear_mess ne %then %do ;
   %let oops= ;
   %let oops2= ;
   %let clear_mess=;
   %readin3 ;
%end ;

%else %do ;
   %if &nml ne %then %do ;
      %let nms=&nml ;
   %end ;
   %if &nm2 ne %then %do ;
      %let nms=&nms &nm2 ;
   %end ;
   %if &nm3 ne %then %do ;
      %let nms=&nms &nm3 ;
   %end ;

%endif ;

%let ok1=y;                                                                # 11

%if %sysfunc(fileexist(&fi)) ne 1 %then %do ;                           # 12
  %let oops=ERROR &fi (Input Dataset) ;
  %let oops2=NOT FOUND AND/OR THE PATH IS INCORRECT! ;
  %let ok1=n ; %goto tester ;
%end ;
%end ;

%tester:                                                                   # 13
%if &ok1 = n %then %do ;
  %startme ;
%end ;

%if &ok1 = y and &alldone eq %then %do ;                                   # 14
  %readin3 ;
%end ;
%mend startme ;

%decide ;                                                                  # 15
endsas ;

%macro decide ;                                                            # 16
  %if &alldone ne %then %do ;
    %end ;
  %else %do ;
    %startme ;
    %end ;
%mend decide ;

# 1 Generate substr code only if needed.
Reading in different external data files we know we may have between 1 and 3 special substr
functions that need to be coded. But, is it 1 or 2 or 3? Once the parameter window has been filled
in the program can determine how many substr functions to generate.
%if &nm1 ne %then %do ;
  &nm1=substr(reference_codes,&st1,&for1) ;
%end ;
Only if something is filled in for &nm1 will the substr code be generated.
# 2 and # 10 Using the variables created by the substr function
Because the program can determine how many variables and the names of the variables created we can use some simple macro code to make a list of names for the Proc Freq.

```sas
%if &nm1 ne %then %do ;
  %let nms=&nm1 ;
%end ;
%if &nm2 ne %then %do ;
  %let nms=&nms &nm2 ;
%end ;
%if &nm3 ne %then %do ;
  %let nms=&nms &nm3 ;
%end ;
```

This code is placed in the parameter window program. A new macro variable will contain the test typed in on the &nm1, &nm2, and &nm3 lines (separated by a space). The new macro variable &nms is used on the tables statement in the Proc Freq

```sas
Proc freq data=saveme.oss_&loc.&dt ;
  tables &nms ;
run ;
```

# 3 The options statement
The important concept here is to turn off all the debugging options. Your end user does not need to ‘discover’ a log file filled with ‘interesting’ messages.

# 4 Appearance of the screen. What the end user sees.
The dm command allows you to change Display Manager options.

```sas
dm 'AWSMAXIMIZE' ;
dm 'wdockview' ;
dm 'log off' ;
dm 'WSTATUSLN off' ;
dm 'WWINDOWBAR off' ;
dm 'zoom' ;
```

AWSMAXIMIZE sets the Application Work Space to maximum.
WDOCKVIEW closes all extra Display Manager windows (Explorer – Results).
LOG OFF closes the log window
WSTATUSLN closes the Windows Status Line
WWINDOWBAR closes the
ZOOM makes the front most window fill the entire AWS
# 5 Initializing the fields displayed on the parameter screen.

 Mostly, I want all the fields to be blank. However, if you do want your users to see a default value – like the path where the data should be stored – you would add it here. Of course, the end user can overtype the default values – changing whatever needs to be changed.

```sas
%let fi=c:\xyz_corp\data;
%let nm1=;  %let nm2=;  %let nm3=;
%let st1=;  %let st2=;  %let st3=;
%let for1=; %let for2=; %let for3=;
%let loc=;  %let dt=;
%let oops=; %let oops2=;
%let nms=;  %let alldone=;
```

# 11 and #12 Does the input dataset exist?

 Whose to say our end user hasn’t made a typo? Why run the macro if the input dataset doesn’t exist? Let’s change our program to check for the dataset and display a message on the screen if it doesn’t exist.

```sas
%let ok1=y;
  %if %sysfunc(fileexist(&fi)) ne 1 %then %do;
    %let oops=ERROR  &fi (Input Dataset) ;
    %let oops2=NOT FOUND AND/OR THE PATH IS INCORRECT! ;
    %let ok1=n ; %goto tester ;
  %end ;
First set &ok1 to ‘y’ (YES everything is OK). Next, what we do is to test the filename typed into &fi. The fileexist function returns a code of 1 if the file exists. Any other code indicates a problem. So, if the return code is “ne 1” set the error message text into &oops and &oops2. Macro variable &ok1 is set to n (NO things are not OK).
```

# 13 Jump down to statement label TESTER.

 Since we have set &ok1 to ‘n’ the %if evaluates to true and we loop back to the top of the macro – redisplay FIRST.

```sas
%tester:
  %if &ok1 = n %then %do ;
    %startme ;
  %end ;
```

# 6 Display messages

 We will use the first two rows of the screen to display messages. Both &oops and &oops2 are initialized to blank (see # 5 above). The first time through the macro nothing appears.

```
"&oops" color=red / "&oops2" color=red
```

But, if we have just come from %tester: that means that there is an error message to display. Here is what we would see:
# 7, #8 and #9 A couple of user friendly touches

The &alldone macro variable will allow users to exit the system without running anything. &clear_mess sets &oops and &oops2 to blank.

```latex
@20 'Exit w/o Running:' +2 alldone 1 color=green required=no attr=underline
+5 'Clear Messages: ' color=blue +1 clear_mess 1 color=green attr=underline
//
@10 '******************** Hit ENTER to run ******************' color=green
```

If &alldone is not blank - %goto tester – ok1 is not blank – so you fall through to %decide without running any further code.

```latex
%if &alldone ne %then %do ;
   %goto tester ;
%end ;
```

If &clear_mess is not blank – blank out the error messages and &clear_mess – run %readin3. This way when the program finishes and FIRST is redisplayed there are no error messages.

```latex
%if &clear_mess ne %then %do ;
   %let oops= ; %let oops2= ; %let clear_mess= ;
   %readin3 ;
%end ;
```
# 14 When everything is checked and OK run %readin3

```sas
%if &ok1 = y and &alldone eq %then %do ;
  %readin3 ;
%end ;
```

# 15 abd # 16 How to keep looping through window FIRST?

A short macro is all we need. The decide macro checks the &alldone variable. If it’s not blank then nothing happened. Else the STARTME macro is executed. The decide macro is called at the bottom of our main program.

```sas
%macro decide ;
  %if &alldone ne %then %do ;
  %end ;
  %else %do ;
    %startme ;
  %end ;
%mend decide ;
```

Making a desktop ICON to invoke our application

This is what you type into the “Target” box in the icon’s properties.

"C:\Program Files\SAS Institute\SAS\V8\sas.exe" -awstitle "Monthly Sales Data" -autoexec 'c:\xyz_corp\source\read_outside_sales_07.sas' -noenhancededitor -nodmsexp

The –awstitle puts the text Monthly Sales Data in the blue bar at the top of the window. -autoexec is where the program lives
- -noenhancededitor and –nodmsexp control which SAS windows are opened.

Conclusions

Our goal was to make an infrequently run, repetitive task user proof. Give the user a desktop icon to click, hide all source code, provide an informative screen with blanks to fill in, and edit check user input. Return context specific error messages and allow for corrections. Now running your program should stay where it belongs – in the hands of the end-user.

Contact Information

George P Sharrard Ph.D.
GPS Corp
14 Sunwich Road
Rowayton, CT 06853
(203) 838-7248
georges00z@aol.com
Acknowledgements

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.