

Transparent Macros

A Simple Way to Add-in Personalized High Quality Re-Usable SAS® Macros in Any Operating System

*By Paul D. McDonald,
SPIKEware, Inc.*

Abstract

This paper shows how to take macros that you have written and integrate them directly into the SAS Software System™ so that they look, act, and feel like part of SAS. Included in this paper are several macro utilities that allow you to do this.

Audience

This paper is written for experienced SAS programmers who are well versed in the SAS Macro Language. System Administrators will also find this paper useful as a reference guide to integrating site-specific utilities to the SAS Software System.

Trademarks and Copyright Notices

The SAS Software System, SAS Macro Language, and other references to SAS products are trademarks of the SAS Institute, Cary, North Carolina.

Microsoft and Microsoft Word97 are trademarks of Microsoft, Inc., Redmond, Washington.

All SAS source code shown in this paper is property of by SPIKEware, Inc. Permission to use, copy, and modify is granted to everyone if the original copyright notice and list of modifications are retained with the code.

Introduction

So you have written some "great" macros and you want others at your company to use them. They save you time and energy! Unfortunately, when you show them to others, they kind of nod their head and go about their way. How do you get people to jump forward?

- Create SAS Macro Add-Ins (including testing, documentation, etc)
- Provide Training to all Users on the Add-Ins
- Encourage users to bring and develop new ideas for SAS Add-Ins

What is an "Add-In" and how is it useful?

The term "add-in" is used regularly by *Microsoft* and is defined as follows by *Microsoft Word97 Help*:

"Add-ins are supplemental programs that extend the capabilities of Word by adding custom commands and specialized features.

You can obtain add-ins from independent software vendors, or you can write your own. To use an add-in, you must install the add-in program on your computer and then load it into Word."

In this definition, Microsoft is talking about add-ins for the popular word processing package *Microsoft Word97*. However, this definition fits for customized SAS Macros.

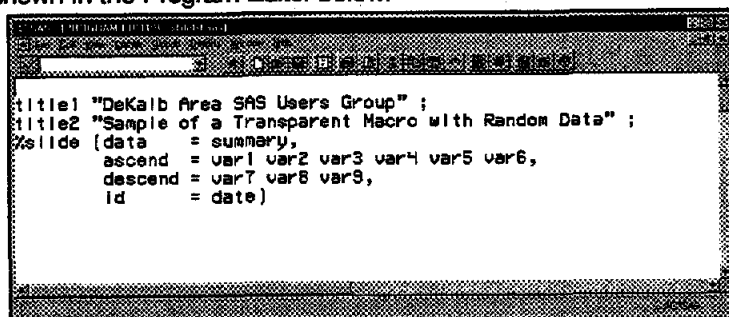
A "SAS Macro Add-In" is a macro that has been thoroughly tested and seamlessly integrated into the SAS Software System in such a way that it appears to be a part of the original release. Add-Ins have the "look" and "feel" of SAS, as well as a documentation reference that fits with the SAS style.

Example

Example Overview

The following example is a report called a "Slide Chart." This macro is called through invoking a SAS Macro called %SLIDE as shown in the Program Editor below.

This macro creates a chart showing relative positions of values in relation to other observations. It is a one-page graphical report, similar to PROC UNIVARIATE (indeed, it uses some of the features of this procedure).



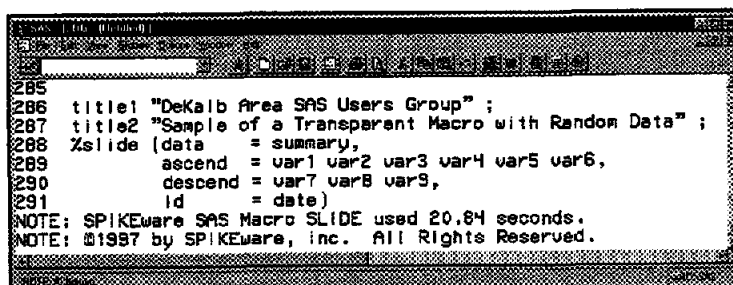
```

title1 "DeKalb Area SAS Users Group" ;
title2 "Sample of a Transparent Macro with Random Data" ;
%slide (data = summary,
        ascend = var1 var2 var3 var4 var5 var6,
        descend = var7 var8 var9,
        id = date)

```

The real "magic" comes in the SAS LOG after the submission of the macro. Notice in the sample log that the macro reports how much time the macro used and called the macro by name. The copyright notice on this macro shows that it is an "add-in" and is not part of the original SAS Software system.

In the sample log, notice that the NOTE: statement for the macro appears just like any other NOTE: from a SAS procedure or data step. For those who work in a color Display Manager, the NOTE: will also match the color of any other NOTE: from the SAS System.



```

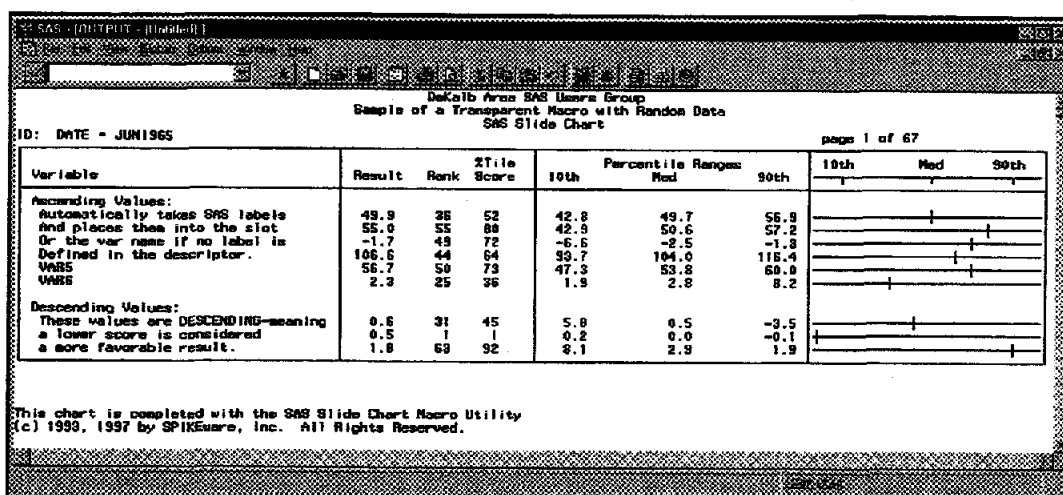
285
286 title1 "DeKalb Area SAS Users Group" ;
287 title2 "Sample of a Transparent Macro with Random Data" ;
288 %slide (data = summary,
289         ascend = var1 var2 var3 var4 var5 var6,
290         descend = var7 var8 var9,
291         id = date)
NOTE: SPIKEware SAS Macro SLIDE used 20.84 seconds.
NOTE: ©1997 by SPIKEware, inc. All Rights Reserved.

```

The output for %SLIDE (shown below) is a bit more complex than the average report. Key points of interest for the report include the following:

- Usage of the variable label in the output listing (a description for each variable)
- Borders for the chart and statistics area

- Graphical indicators of relative position of scores (right column of output)



Example Detail

The next section covers the creation of a SAS Macro Add-In. The sample source code, or code “shell” follows the description of all examples. The steps covered in this section are:

- Initialization and Reset of the SAS Session for an Add-In
- Creation of the INFO:, NOTE:, WARNING:, and ERROR: messages
- Creation of the TIME NOTE:
- Use the correct BORDERS for the correct operating system
- Style considerations
- Cleanup at the end of the add-in

Initialization and Reset of the SAS Session for an Add-In

Before the Add-In can run, it must first set up the SAS Session and capture anything in the session that the macro will change. This often includes the options *NOTES|NONOTES*. In this case, we are not only interested in preserving the options *NOTES|NONOTES*, but the options *NUMBER|NONUMBER*, *LINESIZE*, and *DATE|NODATE* as well.

The *%SYSFUNC* macro function and *GETOPTION*¹ function help us retrieve these values for later use. The combined features can call the values for each of these options and (using a simple *%LET* = statement) store the result in local macro variables. The macro can then change the system options for the running of the add-in and re-set the same options to the original value at the end of the add-in. This makes the running of the macro add-in “transparent” to the user.

For versions before release 6.11 and 6.12E, use the appendix for an alternate approach to retrieving SAS options.

Creation of the INFO:, NOTE:, WARNING:, and ERROR: messages

Using a simple PUT statement inside of a DATA step creates each of these notes. Begin each PUT statement just like the regular statement, with the type of note (in uppercase) followed by a colon. One space after the colon, simply code in your message you want displayed.

¹ See *Macro Facility Enhancements for Release 6.09E and Release 6.11*, © 1996 by SAS Institute, pages 7-13 for a more in-depth discussion of *%SYSFUNC* and *GETOPTION* features.

The PUT 'INFO:' Statement

The PUT 'INFO:' statement will always write to the log. This is true even if the option MSGLEVEL is N (message level set to notes only). If you use INFO: statements, make sure you always want them to be seen.

The PUT 'NOTE:' Statement

If the options NONOTES is on, then the each data step will only display the last NOTE: from the put statements. If there are other PUT statements following the NOTE: put statement, then the NOTE: put statement will not appear in the log. For this reason, it is best to have one DATA _NULL_ for each NOTE: statement. In display manager, SAS will print the appropriate color for each message type.

The PUT 'WARNING:' Statement

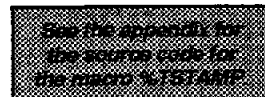
This is the simplest statement of this type. There are no restrictions in the SAS System.

The PUT 'ERROR:' Statement

This statement will set the automatic variable _ERROR_ equal to 1. However, it will not force the SAS System to abend automatically if the *OPTIONS ERRORABEND* is selected. If the ERROR is not "critical," meaning that finding the error means only that the macro will not function properly, it is best to use the %GOTO statement (see "Style Considerations: Errors and Invalid Values" later in this paper). However, if the error is critical to the SAS System, follow the PUT 'ERROR:' statement with an *ABORT ABEND* statement.

Creation of TIME NOTE:

The macro %TSTAMP calculates the start and end time of the macro. When the value of MARK=END, the macro continues with the messaging feature showing the run time of the macro.



Essentially, the current TOD (Time of Day) is stored in a macro variable at the top of the macro. At the bottom of the macro, the recorded value is compared to the current TOD and then the run time can be easily calculated and written to the log using a PUT 'NOTE:' statement.

Use the correct BORDERS for the correct operating system

This is a variation of preserving the system options (as shown above). The use of the %SYSFUNC macro function and GETOPTION function allows the macro to find the value of the option FORMCHAR. Use the SUBSTR function on the result of %SYSFUNC(GETOPTION()) to determine which characters are appropriate for different parts of the output.

When using this feature, it is always a good idea to include a FORMCHAR= option on the macro add-in. This allows for a manual override of the current FORMCHAR= values. Another idea (very useful for ASCII systems) is to hard-code these values using the BYTE function to use specific values. This, however, should be an option for the user and not the default.

Style Considerations

The more that the add-in looks and feels like a SAS procedure or SAS function, the more likely it is that more people will use it. This includes matching "style" or "approach" to the add-in from a user standpoint.

Keywords

Be sure to use the same variable keywords. It is best if these values match not only the add-in macros written for the site, but also match the SAS System. Some of these keywords include:

Input/Output keywords

- DATA =
- OUT =
- INFILE =
- FILE =

Variable keywords

- VAR =
- ID =
- BY =
- FORMAT =

Other Keywords

- OPTIONS =
- LABEL =
- FORMCHAR =
- OBS =

Messages

Write the messages to the SAS LOG concisely and in the same style as a SAS Procedure would write a similar note. For example, the %TSTAMP macro varies only slightly from a regular procedure time stamp. Compare the two statements below:

SAS Time Stamp	NOTE: The PROCEDURE MEANS used 8.08 seconds.
SPIKEware Time Stamp	NOTE: The SPIKEware SAS Macro SLIDE used 20.84 seconds.

Errors and Invalid Values

If an invalid value will cause a problem with the SAS session, it must be checked for a valid value. If an invalid value is used, the %GOTO statement can be used to skip the rest of the code. It is also a good idea to use a conditional PUT 'ERROR:' to inform the user exactly why the macro stopped executing. This is a good application of the PUT 'INFO:' statement, as shown below:

SAS LOG
ERROR: Variable named _TOP_ in dataset WORK.SUMMARY conflicts with reserved name. ERROR: The macro will stop executing. INFO: Suggest renaming variable _TOP_ in dataset WORK.SUMMARY. NOTE: The SPIKEware SAS Macro SLIDE used 3.04 seconds.

Cleanup at the end of the Add-In

Clean-up is the simplest part of all—if you have first set up your system to handle the cleanup! Use PROC DATASETS to delete any data sets, data views, and catalogs created by the macro add-in. Also, be sure to erase any other external files created by the add-in (you can use the X command or other features in the SAS system for this).

Installing your macros as "Add-Ins"

Once you have thoroughly tested and designed macro add-ins, you can install them on the system for everyone to use. This is very simple with the SAS AUTOCALL facility and the AUTOEXEC.SAS program. Follow these simple steps:

- 1) Copy the macros to a file location (read-only preferred) accessible by everyone at your company.
- 2) Use the AUTOCALL facility in AUTOEXEC.SAS to call the macro library. Be sure to include the autocall for SAS Institute macros (usually in ISASROOT)
- 3) Define LIBNAME MACRO 'file-specification' for exclusive use by the utility macros
- 4) Define FILENAME GARBAGE 'file-specification' for exclusive use by the utility macros

AUTOEXEC.SAS

```

filename basemacs 'c:\sas\core\sasmacro' ;
filename prodmacs 'c:\sas\macros\prodmacs' ;
filename garbage 'c:\sas\macros\garbage.txt' ;
libname macro 'c:\sas\macros\garbage' ;

options maautosource sasautos=(basemacs prodmacs) mrecall nonotes ;

data _null_ ;
  put 'NOTE: SPIKEware SAS Macros Package installed (c) 1997 by SPIKEware, Inc.' ;
run ;

options notes ;

```

A few words on documentation

The release of the add-in is not complete until the documentation is complete. Be sure to include the following in your documentation:

- 1) Matching style to the SAS System documentation (contact The SAS Institute for a copy of The SAS Software System Documentation Style Guide)
- 2) Include a usage and reference section
- 3) Include the original source code
- 4) Include references to any macro called by the add-in (if applicable)
- 5) Include samples and sample data, if applicable
- 6) Consider creating a *Microsoft Help* file or *HTML* file for PC use (even if on a mainframe, the HELP files can work on the PC) or even a SAS/AF application

Conclusion

You can write the most robust macros in the world, but if no one uses them, you are not really increasing value. You may be more efficient, but your colleagues do not enjoy the same features. If they do not share in your efficiency and your tools, then you have not truly "built" a system and made a difference. The scientific method calls for us to "Observe, theorize, attempt to prove, and share your findings with others."

The most important part of the scientific method is the last part—"share your findings with others." If you do not communicate with others, you could continue down the "wrong" path forever. Communication among your colleagues is the single most important part of the development of any process and/or community, whether that group be SAS Programmers, a high school football team, boy scout troop, church, city, nation, or family. This communication promotes growth for the entire group as well as the individuals within that group.

Remember to share!

Recommended Reading

- "Developing a SAS System Autocall Library as an Effective Tool Kit," Wilson, Steven, Proceeding (Wisconsin-Illinois SAS Users Group), June 15, 1995
- "Developing Facilities and Tools for SAS Applications," Van Dusen, Philip, Proceedings of the SAS Users Group International (SAS Institute, Inc.), 1995
- "Submitting Macro Language Code in Window Commands," Johson, Martha and Michael Gilman, Observations, (SAS Institute, Inc.) third quarter 1993
- "Structured Design and CASE Tool Use in SAS Macro Systems," Burger, Thomas and Phillip Pochon, Proceedings (Midwest SAS Users Group), 1997
- "Macroitis—A Virus or a Drug," Schrempf, Margaret, Proceedings (Wisconsin-Illinois SAS Users Group), June 15, 1995
- "Macros—The Good, The Bad, and The Obnoxious," Schrempf, Margaret Proceedings of the SAS Users Group International (SAS Institute, Inc.), 1995

Appendix I

Sample Source Code of a SAS Macro Add-In

Description	Source Code
Initialize Macro	<pre>%macro sample (data=, ascend=, descend=, id=, formchar=) /des='Creates graphical rank/%tile report' ;</pre>
Localize variables	<pre>%local data ascend descend id var notes date number ls formchar ;</pre>
Use %SYSFUNC and GETOPTION to pull values	<pre>%let notes = %sysfunc(getoption(notes, keyword)) ; %let date = %sysfunc(getoption(date, keyword)) ; %let number = %sysfunc(getoption(number, keyword)) ; %let ls = %sysfunc(getoption(ls, keyword)) ;</pre>
Change OPTIONS to desired values	<pre>options ls=128 nodate nonumber nonotes msglevel=n ;</pre>
Run %TSTAMP to initiate line stamp	<pre>%tstamp (name=sample, mark=start)</pre>
Use a simple PUT statement to print notes to the log.	<pre>data _null_ ; put 'NOTE: You can put any message you want in here.' ; run ;</pre>
/*your code*/	<pre>/* OTHER SAS STATEMENTS */</pre>
Run %TSTAMP again to print line stamp	<pre>%tstamp (name=sample, mark=end)</pre>
Re-set SAS options	<pre>options &notes &ls &date &number ;</pre>
End Macro	<pre>%mend sample ;</pre>

Appendix II

Macro %TSTAMP source code

```

%*-----*
| SOURCE:   c:\sas\macros\prodmacs\tstamp.sas
|
| PURPOSE:  This macro is used for start and end time stamps. It is often
|           called by other macros and used to mark in the SAS log the run time
|           of a macro, although it may be used in open code for recording the
|           run time of a particular section of SAS code. It is designed to
|           make a standard site macro function more like a SAS Procedure.
|
| USAGE:    %tstamp (name = _name_of_section_of_code_to_test_, <required>
|                mark  = _START_or_END_, <required>
|                owner = _Owner_of_macro_code) <optional>
|
+-----+
| VERSION:  SAS v6.12 / Licensed DeKalb Area SAS Users Group
| SYSTEM:   Microsoft Windows 95 / by SPIKEware, Inc.
| SITE:     DeKalb Area SAS Users Group / 266 Odium Court
|           DeKalb, Illinois / Schaumburg, IL 60194
|           / (847) 798-0729
|           / Paul_McDonald@msn.com
+-----+
| WOOF! (c) 1997 by SPIKEware, Inc. All rights reserved.
|
%*-----* ;

%macro tstamp (name=, mark=, owner=SPIKEware) /des='Records run time of a section of code' ;

%*-----*
| Initialize macro variables and set OPTIONS NONOTES |
%*-----* ;

%global &name ;
%local mark hour minute second note notes owner ;
%let name = %upcase(&name) ;
%let notes = %sysfunc(getoption(notes, keyword)) ;
option nonotes ;

%*-----*
| Condition: &MARK resolves to START, indicating the TOP MARKER |
|           of the time stamp. Assign DATETIME value to the |
|           marker variable. |
%*-----* ;

%if %upcase(&mark) = START %then %let &name = %sysfunc(datetime()) ;

%*-----*
| Condition: &MARK resolves to END, indicating the BOTTOM MARKER |
|           of the time stamp. Calculate the run time of the |
|           marked section and create messages to the SASLOG. |
%*-----* ;

%else %if %upcase(&mark) = END %then %do ;

%*-----*
| calculate run-time values |
%*-----* ;

data _null_ ;

```

```

call symput('hour', hour(datetime()-&&&name)) ;
call symput('minute', minute(datetime()-&&&name)) ;
call symput('second', round(second(datetime()-&&&name), .01)) ;
run ;

%*-----*
| Conditionally create sentence structure |
%*-----* ;

%if &hour = 0 %then %let hour = %str() ;
%else %if &hour = 1 %then %let hour = %str(1 hour) ;
%else %let hour = %str(%trim(&hour) hours) ;

%if &minute = 0 and &hour ne %str() %then %let minute = %str(and) ;
%else %if &minute = 0 %then %let minute = %str() ;
%else %if &minute = 1 %then %let minute = %str(1 minute) ;
%else %let minute = %str(%trim(&minute) minutes) ;

%let note = NOTE: The &owner SAS Macro &name used &hour&minute%trim(&second) seconds. ;

%*-----*
| Use DATA _NULL_ to print note instead of %PUT. This adds the proper color to |
| the message in the log and matches a standard SAS NOTE. |
%*-----* ;

data _null_ ;
put "&note" ;
run ;

%if &owner = SPIKEware %then %do ;
data _null_ ;
put "NOTE: (c)1997 by SPIKEware, Inc. All Rights Reserved." ;
run ;
%end ;
%end ;

%*-----*
| condition: INVALID VARIABLE |
%*-----* ;

%else %do ;
data _null_ ;
put 'WARNING: Invalid input for macro variable &MARK=' "&mark.." ;
put 'WARNING: The macro TSTAMP will not function properly.' ;
put 'INFO: Valid values for &MARK are START or END.' ;
run ;
%end ;

%*-----*
| Reset OPTIONS NOTES to original value |
%*-----* ;

option &notes ;
%mend tstamp ;

%*-----*
| end of macro |
%*-----* ;

```

Appendix III

Alternative to %SYSFUNC and GETOPTION

This method works—sort of. It is dependent on the SAS LOG originally writing to the LOG instead of to an external file. Since this is normally the case, it should not be an issue. If the Log has been re-directed to another file, the log will be diverted back to the SAS Log output. The steps are quite simple:

- 1) Use *PROC PRINTTO* to re-direct the SAS LOG to the file *GARBAGE*.
- 2) Run *PROC OPTIONS*
- 3) Use *PROC PRINTTO* to bring the SAS LOG back to the actual LOG instead of the external file *GARBAGE*.
- 4) Use a *DATA _NULL_* to read in the file and search for the options that you intend to change in the add-in.
- 5) Use *CALL SYMPUT* to assign the initial value of the option to a macro variable
- 6) Run the remainder of the add-in macro
- 7) Re-initialize the options using the macro variable for the option.

The source code looks something like this:

SOURCE CODE

```
%MACRO APPENDIX ;
  PROC PRINTTO LOG=GARBAGE NEW ;
  RUN ;

  PROC OPTIONS ;
  RUN ;

  PROC PRINTTO LOG=LOG ;
  RUN ;

  DATA _NULL_ ;
    INFILE GARBAGE ;
    INPUT @1 OPTION $7. @ ;
    IF UPCASE(OPTION) IN ('NOTES','NONOTES') ;
    CALL SYMPUT ('NOTES', OPTION) ;
  RUN ;

  OPTIONS NONOTES ;

  /* MORE SAS STATEMENTS */

  OPTIONS &NOTES ;
%MEND APPENDIX ;
```