Paper 11887-2016

Considerations in Organizing the Structure of SAS® Macro Libraries

Roger D. Muller, Ph.D, Data To Events, Inc

ABSTRACT

SAS® macros offer a very flexible way of developing, organizing, and running SAS code. In many systems, programming components are stored as macros in files and called as macros via a variety of means so that they can perform the task at hand. Consideration must be given to the organization of these files in a manner consistent with proper use and retention. An example might be the development of some macros that are company-wide in potential application, others that are departmental, and still others that might be for personal or individual user. Super-imposed on this structure in a factorial fashion are the need to have areas that are considered: (1) production - validated, (2) archived - retired and (3) developmental. Several proposals for accomplishing this are discussed as well as how this structure might interrelate with stored processes available in some SAS systems. The use of these macros in systems ranging from simple background batch processing to highly interactive SAS and BI processing are discussed. Specifically, the drivers or driver files are addressed. Pro's and con's to all approaches are considered.

INTRODUCTION

This paper addresses approaches for the use of SAS macro code in systems that involve more than one user. Individual users who are using macros for “one-off” tasks do not face nearly as many issues as do developers making systems that will be shared and/or will be used long-term. Far too often, projects start as a single-focused endeavor with no thought to long-term use. When the projects move out of a single-user or limited number of users arena, it is time to sit back and discuss the more global future.

Above all else, the following are most important considerations:

- The coding must produce <u>accurate</u> results. It must do what it says it will do and do it correctly.
- The code must be <u>maintainable</u>. WRITE THE CODE SO THAT SOME FUTURE PROGRAMMER NEW TO THE SCENE CAN USE IT AND DO MAINTENANCE WORK. Failure to do this can tarnish your image, your department’s and management’s images, SAS systems in general and more. Rebuilding systems is expensive.
- The code must be simple to understand. As projects evolve, time should be taken to sit back and look and what has happened and what needs to happen rather than just “plowing ahead” and slapping more code on top of code that should be reworked. It may be time to “un-complex” the project.
- The code must contain appropriate (usually generous) number of comments.
- There must be ancillary documentation. The location of this documentation should be well known by all programmers.
- The code should not be “cryptic”. Variable names should represent the project at hand. If working with short variable names, make sure that labels accurately reflect what the variable is about. Use SAS variable names like lab_test instead of lt.
- All variables should have labels.
- Try to use longer variable names if permissible.

What should be either dismissed or minimized in macro-driven systems:

- Cleverness. If the next programmer working with the code can't immediately grasp what is going on, bad things will probably happen. Nobody cares how clever you are. They want code that is correct and maintainable.
- Worrying too much about "machine efficiency, storage capacity, etc." While these considerations do deserve consideration, the continual decrease in the cost of hardware and storage over the years can permit for some leniency here.

The thought processes presented behind the concepts in this paper were accumulated over a long period of SAS software usage in a number of organizations. This experience started prior to the introduction of the macro facility. As the macro facility has been expanded to include many new features that streamline its use over the years, many programmers have not taken advantage of these featues.

The more recent introduction of Stored Processes in the SAS BI environment is an extension of macro usage. The use of Stored Processes will not be addressed in detail in this paper since they are not yet "mainstream" in many organizations.

HOW TO "DRIVE" MACRO-BASED SYSTEMS

For years the term "driver file" or "driver program" is thrown about by SAS programmers and SAS teams. Surprisingly, very little has been published about driver files. Typical use in organizations has been to find an old driver file, copy it and try to enhance it.

A Google search for "SAS driver programs" (without quotes, or assorted variations of these words) does not produce a huge listing of links. Most returns are for printer drivers, ODBC drivers, database drivers etc. A few of significance that do pertain to controlling program execution include:

1. Basics of driving a system,Camishev, 2011. http://support.sas.com/resources/papers/proceedings11/098-2011.pdf. This paper is an excellent overview of the process.
2. A more advanced discussion of creating a macro driver file, Minor, 2013. http://www.pharmasug.org/proceedings/2013/CC/PharmaSUG-2013-CC34.pdf. This paper goes so far as having an email component to notify the submitter of program execution results. This paper addresses the use of drivers in a variety of SAS environments (interactive PC, batch server, etc).
3. A discussion of assigned macro variables in a driver program, Fecht and Stewart 2008. http://www2.sas.com/proceedings/forum2008/164-2008.pdf/
4. Stored Processes. Stored Processes are in reality SAS macros that are developed in the SAS BI Archictedure. Eberhardt, 2008 ,wrote an introduction to writing stored process that explains in detail the environment needed to do this. http://www2.sas.com/proceedings/forum2008/035-2008.pdf. Again, since stored processes and the associated use of prompts, the SAS Add-In for Microsoft Office, etc. are not mainstream yet in most organizations, they will not be further addressed. They are certainly a means of controlling (driving) macro-based systems.

WHAT APPROACH SHOULD SAS DRIVER FILES TAKE TO INCLUDING OTHER SAS CODE?

There are essentially two techniques used in driver files by which other files containing SAS code may be brought in and ran:

1. %include filename. This has been around for a long time and precedes the introduction of SAS macros. There is absolutely nothing wrong with using this technique yet today. Some would maintain that this technique provides the easiest way of maintaining and understanding SAS code written by other users.
2. The SAS autocall macro capability. By searching a predefined path of directory locations, SAS will run the first occurrence of a called macro in that path.

CONTENTS OF A DRIVER FILE

The following discussion is very general in nature. Desired content of a driver file may vary, but these are reasonable starting points.

WHAT SHOULD BE IN A DRIVER FILE?

1. Lots of comments as to purpose, ownership, use, dates, etc.
2. Try to set the assorted system macro options to on or off here. Avoid turning these on/off in programs that are called later as they can be buried and difficult to find. If this was done in "production" files, you may not even have write access to them to change settings,
3. Macro variable assignments that are passed into macros later.
4. %includes for filenames that are to be brought in and processed. These may or may contain macro definitions
5. Definition for macro autocall search order (sasautos). This is very important – see later discussion.
6. Possibly some high-level titles and footnotes. This may be the best place to globally set them and maintain them.
7. Turn on option mautolocdisplay if sasautos are used. This displays the location of the file that contained the code for the macro that was called. Remember that SAS picks the first location of macro in a file according to the search order established by sasautos. Debugging is impossible if you are looking at code in the wrong location.

WHAT SHOULD NOT BE IN A DRIVER

1. SAS code that "clutters up" the driver that could and should have been in a called macro or in a file that is %included later.
2. Anything that does not "direct the flow" or set values for macro variables.

GOALS FOR SYSTEM DESIGN

1. All programs are run by a driver file, regardless of the SAS environment.
2. Development is done in development folders.
   a. Core programs contain code and SAS macros that are intended to require no modification. They are too be left alone in as much as possible to avoid "flooding" the folders with a multitude of copies containing only minor code differences.
   b. Supplemental programs contain SAS code and SAS macros that require frequent modifications. They are called into the program flow either from the driver file or from core programs.
   c. The development area files are modified, duplicated, renamed etc. by the programmer.
3. The production area contains the current production version of the programs. There are both "Core" and "Supplemental" files. The moving of files into the production area is the responsibility of the system administrator.

SCOPE CONSIDERATIONS ON THE ACTUAL SAS CODE FILES

How do we manage the actual SAS files that contain macro code (or for that matter – nonmacro code).? Let's consider 2 extremes.

- At one end of the spectrum we are dealing with a single programmer developing some sort of system for his/her own use. There is no issue with multiple users, all code is kept in his own desired locatons.
- At the other end of the spectrum is a system that is used by hundreds, maybe thousands of programmers/users. The issues here very rapidly become things like security, version control, updates, archives, etc. Organizations with these type of requirements are likely to develop their own systems for checking files in and out, backing up, etc. These subjects are not discussed in detail in the paper.

In between we may have smaller departmental groups with requirements falling in between the extremes.

THE 2X2 MATRIX FOR SAS MACRO CODE SYSTEMS

A 2x2 matrix is proposed for the development and production architecture. The components are a “core code” folder and a “supplemental code” folder. There are folders devoted to “development” and those devoted to “production”. See Table 1. Proposed 2x2 Matrix Structure for SAS . This table also shows the proposed search order for options sasautos= to search for macro names.

Programmers develop code in the “development” folders where they have complete read and write access. Programmers request administrators to move code to the production folders since only administrators have write access.

Another dimension to folder grouping may occur in the Production Area. There may be an “overall” set of folders and then there might be levels of folder devoted to something like “Project”, “Reporting Effort”, “Department” etc and named accordingly. Remember that these are the permanent long-term depots and must be carefully established and maintained.

A more extensive explanation of this structure is in Table 2. Files Associated With a SAS Macro System.

Table 1. Proposed 2x2 Matrix Structure for SAS Folders

	Development	Production
Core Code	Development Core (2 or 3)	Production Core (4)
Supplemental Code	Development Supplemental (1)	Production Supplemental (2 or 3)
Numbers in cells refer to the most likely search order to be used in options sasautos=. Logic is: (1) New code in development most likely supercedes older code in production. (2) macros in Supplmental code may need to be called by Production Code so they must be available.		

CONTENTS OF FILES

Remember that these files are in “folders” or “subdirectories” in your system as outlined above. These are files name *.sas and contain SAS code.

WHAT SHOULD “CORE CODE” FOLDER FILES CONTAIN?

Header/banner (name, purpose, date, creator, notes on use, etc ---- all in SAS comments)

A current version number of the file in a macro variable. Example %let filevers=3.7; *this is used to print a version number to the log, output, etc.

A macro variable with the current file name to be printed to the log, listing, etc. %let curfile=mysimple_example;

Essentially the SAS code and the code for macros that is considered to be bulk of the system.

WHAT SHOULD THE “SUPPLEMENTAL CODE”FOLDER FILES CONTAIN?

As with the core code, the header/banner, macro variables for current version, filenames, etc.

Lines of SAS code (almost always in a macro) that are called from the “core code” file to do things that occasionally need modification. They may or may not always be used.

Again, the purpose of the supplemental code files is to avoid “cluttering” up the “core code” files with too many file versions that are nearly all the same except for a few minor tweeks. The “supplemental code” files provide a means of slipping in code to the “core code” processes.

WHAT SHOULD FILES IN THE PRODUCTION FOLDERS CONTAIN?

These are files that have been fully developed to the programmer’s best knowledge and are ready to “go live”. Note that there may be folders that are very global in nature and there may be others specific to some reporting effort, project, department. A full discussion of this is not feasible within the scope of this document. See the accompanying video for some details.

WHAT SHOULD FILES IN THE DEVELOPMENT FOLDERS CONTAIN.

These files are under development by the programmer or programming team. They have full read and write access to them. Obviously these are files that are eventually moved to the production area.

What should the search order be for the sasautos=option? This is specified in the footnote in Table 1. Proposed 2x2 Matrix Structure for SAS . The logic here is that during the program development phase, the code under development is what is being tested so it should be searched for new macros first.

OTHER QUESTIONS

Question: are “production core code” files located in “report specific” folders or in more general “master system” folders. They could be in either.

Question: What about “supplemental code” files. Are they in master areas or in report specific areas? They too could be in either place.

Question: How does SAS decide what is supposed to be called? This is determined by the search order established in options sasautos. SAS will use the first occurrence of that macro code as the one to be use. A thorough understanding of this concept is essential to making the system work.

Question: Are “supplemental code” macros called from (a) drivers, (b) “core code”, (c) “supplemental code”, or (d) any of these choices. The answer is (d).

Question: How do I know for sure what was called and what location SAS used as the source? By turning on option mautolocdisplay. NEVER TURN THIS OFF IF SASUTO CAPABILITY IS BEING USED. This reports back the name of the file and location from which the macro was compiled. (see example later).

Table 2. Files Associated With a SAS Macro System.

	Driver File	***Developmental Files***		***Production Files***	
		Core	***Supplemental***	***Core***	***Supplemental***
Definition and purpose	*This file contains SAS code that points to the files to be executed. It drives the processes for the "current run". It may contain macro calls.*	*New code under development. May call supplemental code*	*Code under development to supplement either core code under development or core code in production*	*Code that is considered done and in production*	*Complete code that supplements core and is usually called in by macros from core production or core development folders*
How is it executed ?	*Programmers and users submit the driver file to run.*	*Executed by the driver file or onother macro*	*Usually executed by some other macro, may be executed by the driver file*	*Executed by the driver file or by another macro*	*Usually executed by some other macro, may be executed by the driver file*
What does it contain? How is it called and ran?	*1. A Header Comment Area* *2. %include Statements to Include Specific Files of SAS Code and run them* *3. Macro Calls to Developmental or Production Macros* *4. Macro Calls to Supplemental or Core Macros* *5. Macro Variable Assignments* *6. Setting options for SAS macro processing and for general SAS options. Note that sasautos are defined here. Note: Note that macros being called here by the sasautos option must be setup according to very specific guidelines. See Muller, 2014 http://support.sas.com/resources/papers/proceedings14/1862-2014.pdf*	*Simple SAS code* *One macro in each file (enables use of sasautos)* *Multiple macros (in which case the SAS file must be "%included")*	*Simple SAS code, macros, pieces of macros, etc.* *Must either be "%included" or pulled in via sas autos. Depending on content, can be pulled in by the driver file, a Core Development file, a Core Production File, or another Supplemental file.*	*Simple SAS code* *One macro in each file (enables use of sasautos)* *Multiple macros (in which case the SAS file must be "%included")*	*Simple SAS code, macros, pieces of macros, etc.* *Must either be "%included" or pulled in via sas autos. Depending on content, can be pulled in by the driver file, a Core Development file, a Core Production File, or another Supplemental file.*
Who writes code	*Programmers and end users*	*Programmers*	*Programmers*	*Administrators with very restricted write access*	*Administrators with very restricted write access*

CONSIDERATIONS

In an ideal system, the “Core Production” code and macros would not need modification. There are times when small enhancements must be brought in for specific requirements of a reporting effort or some other effort. An example might be that a request was made to strict a report to male subjects only. A simple “if” statement could be called in via a supplemental code call. The alternative approach would possibly require extensive rewriting of the system.

OPTIONS SASAUTOS= AND SEARCH ORDER

The use of the sasautos= option is discussed in Muller 2014 (*http://support.sas.com/resources/papers/proceedings14/1862-2014.pdf*}. Each macro must be in its own file with the same name. This can result in a large number of files. If this is of concern, reverting to the “%include filename” technique can be useful.

Of particular note when using sasautos is the ability to control the sort order for the locations where SAS will look for macros. If you want to look in development libraries first and then look in production libraries, this is feasible.

Some sample code to do this follows. Programmers must remember to turn on the option mautosource to enable SAS to utilize the sasautos functionality.. Note that the last reference is made to sasautos which is the location of the 16 macros furnished with thebase SAS installation:

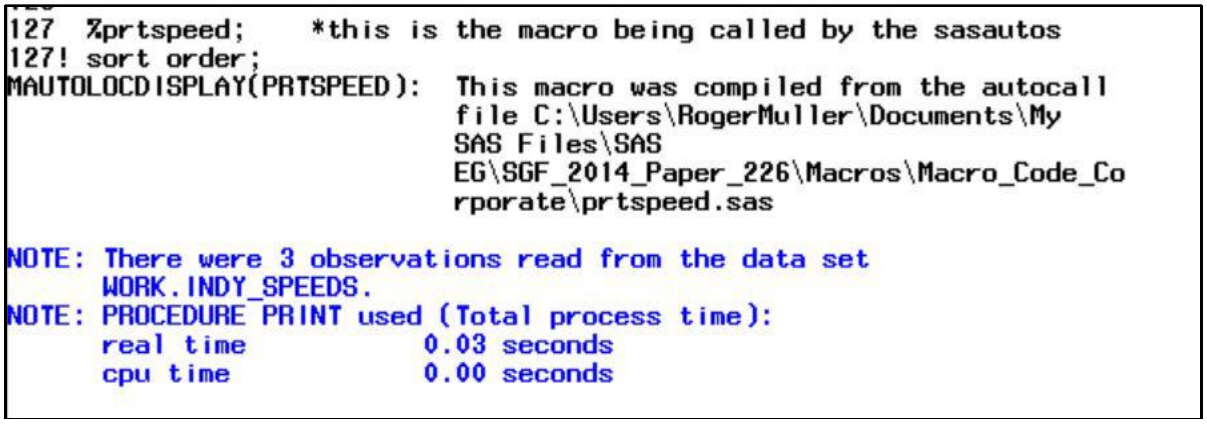
```
Options mautosource;

options sasautos=('C:\Users\RogerMuller\Documents\My SAS
Files\Project_A\Development\Supplement\’
'C:\Users\RogerMuller\Documents\My SAS Files\Project_A\Development\Core\’
‘z:\Master\ResearchDivision\Production\Supplement\’ ‘z:\Master\Research
Dvision\Production\Core\' sasautos);
```

CONFIRMING THE LOCATION OF A MACRO SAS ACTUALLY USED

There is a little-known option in SAS that will tell you exactly what the location of the sasuto macro was. IF THE SASAUTO CAPABILITY IS BEING USED, MAUTOLOCDISPLAY SHOULD ALWAYS BE TURNED ON. There is no reason not to have it turned on as the information returned to the log is compact, extremely informative and easy to read. In this example taken from Muller 2014, a macro prtspeed was located in 3 different directories. The search order was established via sasautos. When the macro was called, the following was returned to the log file.PRETTY SLICK!

Table 3. Information Returned to Log by Option Mautlocdisplay



```
127  %prtspeed;    *this is the macro being called by the sasautos
127! sort order;
MAUTOLOCDISPLAY(PRTSPEED):  This macro was compiled from the autocall
                            file C:\Users\RogerMuller\Documents\My
                            SAS Files\SAS
                            EG\SGF_2014_Paper_226\Macros\Macro_Code_Co
                            rporate\prtspeed.sas

NOTE: There were 3 observations read from the data set
      WORK.INDY_SPEEDS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.03 seconds
      cpu time            0.00 seconds
```

ARCHIVING

1. Old files are moved to the archive area by the system administrator and the files are renamed with the version number appended to the filename.
2. Driver files are also archived by the system administrator, again with a version number.

BACKUP OF DEVELOPMENT AND PRODUCTION FILES

Backup of all files is the responsibity of the system administrator. In most cases, this will be responsibility of the administrator for the server and occurs independently on a nightly basis.

ROLLBACK. Occasionally it is desired to rollback a development file to a previous version during the course of a day's activities. SAS Enterprise Guide provides such a rollback facility in version 7.1 (Hemedinger 2014, Muller 2014, Muller 2015) for embedded sas code files. For external stored files, it is necessary to use a Git program, Hemedinger 2012, or a tool such as Subversion (Cheung and Borowiak, 2015).

CONCLUSION

This paper summarizes some considerations to be given to the organization of SAS code files for building macro driven systems. Code development should be done in developmental folders and moved to production folders by administrators. Within the development and production folders, there should be a "core" code area and a "supplemental" code area. The core area contains the bulk of the programming code while the supplemental area contains those small tweaks that are often necessary to make modifications for minor issues. The goal is to minimize the copying and modifications of the core files and keeping the numbers of near duplicate core files down while still having the flexibility to provide for minor but important adjustments. Specifications for driver files are also presented in a simple orderly fashion.

REFERENCES

1. Camishev, 2011. How to Drive Your SAS® Programs on Cruise Control, http://support.sas.com/resources/papers/proceedings11/098-2011.pdf. This paper is an excellent overview of the process.
2. Cheung, Oscar F. and Kenneth Borowiak, 2015. Pharmasug 2015 Poster PO09. Adding Subversion Operations to the SAS Enhanced Editor. Available at: www.pharmasug.org/proceedings/2015/PO/PharmaSUG-2015-PO09.pdf
3. Minor, B., 2013. Using SAS Driver Programs to Automate Workflows and Respond to the Unexpected. Available at http://www.pharmasug.org/proceedings/2013/CC/PharmaSUG-2013-CC34.pdf.
4. Fecht M and Stewart 2008. Are your SAS programs running you. SAS Global Forum 2008, paper 164. Available at http://www2.sas.com/proceedings/forum2008/164-2008.pdf
5. Eberhardt,P, 2008. Crossing the Border: Steps to Writing SAS© Stored Processes Available at: http://www2.sas.com/proceedings/forum2008/035-2008.pdf.
6. Hemedinger, Chris, 2012. Using source control management with SAS Enterprise Guide. Available at: http://blogs.sas.com/content/sasdummy/2012/10/29/scm-with-sas-eg/
7. Hemedinger, Chris, 2014. New SAS programming features in SAS Enterprise Guide 7.1 Available at: http://blogs.sas.com/content/sasdummy/2014/10/12/eg-71-new-programmer-features/
8. Muller, Roger D., Assorted Videos on SAS Topics Including SAS Enterprise Guide, Macro Systems and More .. Available at: SAS Video Playlists by Roger Muller (Google in YouTube "Roger Muller Youtube SAS Channel" without quotes).

9. Muller, Roger D., 2014, Managing the Organization of SAS® Format and Macro Code Libraries in Complex Environments Including PC SAS, SAS® Enterprise Guide®, and UNIX SAS , available at: http://support.sas.com/resources/papers/proceedings14/1862-2014.pdf (has a video component).
10. Muller, Roger D., 2014, File Management and Backup Considerations When Using SAS® Enterprise Guide (EG) Software. Available at http://www.mwsug.org/proceedings/2014/RF/MWSUG-2014-RF05.pdf.
11. Muller, Roger D., 2015. Making Historical Versions of SAS® Code While Developing in Enterprise Guide 7.1. Available at: http://www.mwsug.org/proceedings/2015/RF/MWSUG-2015-RF-16.pdf.

CONTACT INFORMATION

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

Roger D. Muller, Ph.D.
Data-To-Events, Inc.
317-985-0132
Roger.Muller@Data-To-Events.com
WWW.Data-To-Events.Com