

SAS® Macro Autocall and %Include

Jie Huang, Merck & Co., Inc.
Tracy Lin, Merck & Co., Inc.

ABSTRACT

SAS® provides several methods to invoke external SAS macros in a SAS program. There are two that are most often used, one is the autocall library, which enables users to invoke a macro that is not defined in the same SAS program, while the other is the %include statement, which identifies an entire external file that the user wants to bring into the program and execute immediately. Each method has its' own advantages and disadvantages. It is important to understand how and where to use them properly and effectively. This paper will outline the different features of each method so that users may choose the best approach for their application in the Microsoft Windows environment.

INTRODUCTION

Users can set one or more directories as "autocall macro libraries", which means that if you invoke a macro that has not yet been defined in user SAS session, SAS will search for a file with the macro name in each of the directories listed. Any valid SAS macro can be stored in the autocall library, which can be used by any SAS program or session as needed, and by multiple SAS programs and users at the same time.

Users also can use %INCLUDE to include a specified external SAS file. Thus, users don't need to copy and paste the macro code in every program that invokes the macro.

In most cases there is no difference between these two methods in terms of the way to invoking macros. However, because SAS processes them differently, each approach presents it's distinct feature in different settings.

WHERE TO USE

Users can set the SASAUTOS system option when starting SAS, or use it in an OPTIONS statement during a SAS session. However, be aware that autocall libraries specified with the OPTIONS statement override any previous specification.

%INCLUDE can be used anywhere in a program when users want to bring external SAS files or macros into a current SAS program. This paper will only be discussing "include" SAS macros.

HOW TO USE

Using directories as autocall macro libraries

A user can designate one or more directories as a SAS autocall library to store macro source code. In most cases each macro file in a directory must contain a macro definition with a macro name that matches the filename.

To use the autocall library, system option MAUTOSOURCE must be set to turn on the autocall facility, and SASAUTOS needs to be assigned by specifying each of the autocall libraries.

The syntax of system option SASAUTOS is shown below:

```
SASAUTOS=("library-specification-1"...<"library-specification-n">);
```

Here "library-specification-1"...<"library-specification-n" specifies one or more valid pathnames or environment variables that are associated with pathnames. Note that a pathname is only associated with a directory or a subdirectory. Pathnames must be enclosed in quotation marks if it contains spaces. The value for library-specification must be resolved to a valid pathname. To specify more than one autocall library, users may enclose each of the specifications in parentheses and separate them by either a comma or a blank space. Autocall libraries are searched in the order that users specify them. As shown in the example below, SAS will search for macros the user defined in the directory of "c:\AB001\maclib" first, the directory of "c:\AB001\prot001\maclib" second, and macro library supplied by the SAS system last when a macro is invoked.

```
options MAUTOSOURCE
      SASAUTOS=(c:\AB001\mac lib
                c:\AB001\prot001\mac lib
                c:\AB001\prot002\mac lib
                sasautos);
```

%INCLUDE statement

Users can also use %INCLUDE to include a macro stored in an external file. Shown below are the syntax and sample code,

```
%INCLUDE source(s) </host-options>;
```

```
%include 'c:\maclib\flag.sas' /recl=1000;
```

Here *source* describes the location of the macro that the user wants to access. And *host-options* can include operating environment information. The operating environment can support various options for the %INCLUDE statement with a forward slash (/) preceding the options list. For example, one may use "LRECL" under the Windows operating system. The option, LRECL= *record-length*, can be very useful when including an external SAS file. It specifies the record length (in bytes). This option will be discussed below.

Even when users decide to utilize %INCLUDE to bring in all user defined macros, SASAUTOS should still be used as one of the options. Failure to do so will make all SAS built-in macros unavailable, such as %CMPRES, %DATATYP, %LEFT, and %TRIM etc.

COMPARISON/LIMITATIONS

I. Compilation and Execution

Autocall – The macros will not be compiled and executed until they are invoked for the first time. However, on successive calls, the SAS session will bypass the compiling process. Note, if a macro is redefined, the user should resubmit the new macro definition or close the SAS session and re-open it. Otherwise SAS will re-execute the originally compiled code and ignore any changes made.

%INCLUDE – The macros will be compiled everytime they are included by %INCLUDE and will be compiled for each %INCLUDE statement. When using %INCLUDE to bring in a macro, it brings in the macro definition itself so that there is no issue as with the AUTOCALL method when a macro is redefined.

If a macro is very large or complex, it will take more time to locate and compile when it resides in the AUTOCALL library than using %INCLUDE.

II. Macro, File and Path Names

Autocall – A macro must be named the same as the name of the file it is stored in when it is set up, even though the SASAUTOS option only tells SAS where to look for the macros rather than referencing the macro source code directly.

%INCLUDE – Users need to specify the physical name and path of the external file that is enclosed in quotation marks. Below are two samples of code:

```
Filename macrodir 'c:\maclib\globtxt.sas';  
%include macrodir;
```

A user can also point to individual macro directly by using the following %INCLUDE statement syntax.

```
%include 'c:\maclib\globtxt.sas';
```

The greatest disadvantage of using %INCLUDE is the need for tracking each individual file location, while the autocall library has the ability of locating all programs defined as macros in the directories available.

III. Debugging

The autocall facility is less useful during program development, which is when error checking and debugging occur. Since users need to retrieve and replace code often, the autocall facility provides less convenience for the same overhead. In fact, an updated macro would need to be resubmitted or the SAS session closed in this case. Since program development often focuses on discrete sections of an application, there is less need to have all utility macros available at one time. Also, using concatenated lists of autocall libraries is not as great a convenience if one only needs a handful of macros from any given library.

During the program development, the more efficient method is to use the %INCLUDE statement to retrieve macros during a SAS session.

IV. Maintenance and Version Control

By using autocall libraries, all macro source code could be in one location which means faster updates and better consistency across all programs. Therefore, it is recommended to save all autocall macros in the same directory for easy maintenance and retrieval. However, SAS allows users to direct it to multiple directories. The sample shown earlier is an example of how this could be accomplished. Be aware of the cases when one macro appears more than once within a list of directories. Since SAS searches in the order listed in the SASAUTO, the macro SAS finds first may not be the one or version you intend to retrieve.

TIPS for USING %INCLUDE

From our experience, one important option for %INCLUDE is LRECL=record-length, which specifies the record length (in bytes). Under Windows, the default is 256 bytes, the value of *record-length* can range from 1 to 1,073,741,827 (1 gigabyte) in SAS version 9.

When users use %INLCUDE to include a macro, they may get some unexpected results, and encounter very difficult debugging, especially if macro source code does not appear in SAS log. In this case, we suggest to check the length of each line of the %INCLUDE included, or simply to add the LRECL option to the %INCLUDE statement to extend the

record length, and then re-submit the SAS program. It is possible that source code has been cut off because the default length is not long enough.

Shown below are a simple macro and call programs that could demonstrate points stated above.

```
%macro example;  
    %let yy=The autocall facility is relatively less useful during  
program development, which is when error checking and debugging occur.  
Since users need to retrieve and replace code anyway, the autocall  
facility provides less convenience for the same overhead. Since program  
development often focuses on discrete sections of an application, there  
is less need to have all utility macros available at one time.;  
    %put &yy;  
%mend example;
```

- I. Example1 is the macro call without any options. We got unexpected result with no clue in the log.

```
options;  
%include "U:\paper\example.sas";  
%example;
```

SAS log:

```
options;  
%include "U:\paper\example.sas";  
WARNING: Truncated record.  
ERROR: Macro keyword PUT appears as text. A semicolon or other delimiter may be missing.  
%example;
```

- II. Example2 is the macro call using source2 option which causes the SAS log to show source statements that are being included. We did not get correct result either.

```
options source2;  
%include "U:\paper\example.sas";  
%example;
```

SAS log:

```
options source2;  
%include "U:\paper\example.sas";  
NOTE: %INCLUDE (level 1) file U:\paper\example.sas is file U:\paper\example.sas.  
+%macro example;  
+    %let yy=The autocall facility is relatively less useful during program  
development, which is when error checking and  
!+debugging occur. Since users need to retrieve and replace code anyway, the autocall  
facility provides less convenience  
!+for the same overhe  
+    %put &yy;  
ERROR: Macro keyword PUT appears as text. A semicolon or other delimiter may be missing.  
+%mend example;  
NOTE: %INCLUDE (level 1) ending.  
%example;
```

III. Example3 is the macro call using lrecl option with %include to get the expected result.

```
options source2;  
%include "U:\paper\example.sas" /lrecl=1000;  
%example;
```

SAS log:

```
%include "U:\paper\example.sas" /lrecl=1000;  
NOTE: %INCLUDE (level 1) file U:\paper\example.sas is file U:\paper\example.sas.  
+macro example;  
+    %let yy=The autocall facility is relatively less useful during program  
development, which is when error checking and  
!+debugging occur. Since users need to retrieve and replace code anyway, the autocall  
facility provides less convenience  
!+for the same overhead. Since program development often focuses on discrete sections of  
an application, there is less need  
!+ to have all utility macros available at one time.;  
+    %put &yy;  
+%mend example;  
NOTE: %INCLUDE (level 1) ending.  
%example;
```

The autocall facility is relatively less useful during program development, which is when error checking and debugging occur. Since users need to retrieve and replace code anyway, the autocall facility provides less convenience for the same overhead. Since program development often focuses on discrete sections of an application, there is less need to have all utility macros available at one time.

CONCLUSION

Though both the autocall library and %INCLUDE can invoke macros not physically defined in the current SAS program, each approach has its' own distinct features and may better be used in different settings. %INCLUDE statement can be used whenever needed without requiring setting up an autocall library first. Also, it is simpler to use during program development. Autocall library is the best way to store well established macro source code, such as utility routines or shared macro(s) across protocols, which enables easy maintenance and better consistency when all macros are placed in one directory.

TRADEMARKS

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 trademarks of their respective companies.

REFERENCES

SAS Institute Inc. SAS® Guide to Macro Processing
SAS Institute Inc. SAS® Language: Reference

ACKNOWLEDGMENTS

We would like to acknowledge and thank our managers Richard Lowry and Hong Qi for their support, and comments regarding this paper.

CONTACT INFORMATION

Jie Huang
Merck & Co., Inc.
351 Sumneytown Pike
North Wales State: PA ZIP: 19454
(267) 305-5541
Jie_huang4@merck.com

Tracy Lin
351 Sumneytown Pike
North Wales State: PA ZIP: 19454
(267) 305-8202
trace_lin@merck.com