Paper 036-31

Xamining the X Statement (and Some Other Xciting Code)

George J. Hurley, The Procter and Gamble Company, Cincinnati, OH

ABSTRACT

The X statement has been briefly documented for years in the SAS® Language Guide. However, the usage of this statement in code appears uncommon and its power is underutilized, especially by the novice programmer. This statement can be used to issue operating system commands during an interactive SAS session, consequently allowing for better software integration. The only part that is needed is Base SAS! This short paper will help the reader understand how, when, and why to use the X statement, as well as what type of code works well with it. Furthermore, it will provide sample code. The focus will be on the Windows environment, however much of the paper can be utilized in other operating systems.

INTRODUCTION

The X statement, as described in the Version 6 SAS documentation is intended to, "issue a host command from within a SAS session when you run SAS in display manager or interactive line mode." (SAS Institute, 1990) Unfortunately, the documentation of this statement is not much better in the Version 8 or SAS9 documentations, the latter of which states, "specifies an operating environment command that is enclosed in quotation marks." (SAS Institute, 2005) The description in Version 6 documentation summarizes the power of this statement, perhaps without the reader ever realizing it. More recent references to the X command state the same thing, except it can be executed from the SAS Display Manager command line. So, what does issuing a host command mean? To the novice user, this may not be obvious. In one sentence, it allows you to use any operating system (OS) command from within an interactive SAS session. For the purposes of this paper, the usage of this command from within a Windows environment will be discussed. However, the discussion (but not necessarily the syntax of the OS commands) generalizes to many other operating systems.

I STILL DON'T KNOW WHY I WOULD WANT TO USE THIS

Why would a user want to use this command? Why is it worth one's time to read on? Let us take a scenario. A task has been assigned that involves generating a report for the ubiquitous "management". This report is a specific Microsoft Word document that contains results that can be easily generated using the power of SAS. This document requires a degree of manipulation in Word. This manipulation may seem trite to a programmer, but to all-knowing "management" the form brought about by this manipulation is the "only possible format that could easily understood by the end-user." All jokes aside, every programmer, regardless of profession, employer, or expertise, has been given instructions to produce a standard report in a format that is either not compatible with SAS or would take some very intensive use of proc template to get it close to that format. However, now assume one is well versed in an object-oriented language, like Visual Basic. It is very simple to manipulate Word documents from Visual Basic. If you know the X statement, you write a short Visual Basic program, structure your SAS program cleverly, and execute your VB program from SAS. Problem solved!

Consider the following scenario. Your company uses a software package to generate a particular report. It expects standard input and produces a graph having certain characteristics. Creating this particular graph is the standard operating procedure for your company. Being supportive of the company's rules and cognizant of the fact that the code already exists to generate these graphs using this other software, you seek to produce this graph. However, this software is not as friendly with data manipulation as SAS. It is time intensive to take raw data and massage it into the format necessary to produce these graphs from this software. What is the time savvy programmer to do? Use the X statement! Set up a SAS program that generates the necessary data in a file that is readable by the other software package, but is already in the format required. Then batch submit the other software from SAS, having written the minimum additional code in the other software to read in your SAS data set and/or text files.

Following is another scenario. A coworker has conducted two surveys, one for females and one for males. This coworker conducted these surveys electronically on a handheld device. He has generated copious files that summarize each person's survey responses. However, the two surveys were for different products and had different questions. For simplicity, it will be assumed that the formats are similar; however, this is not a necessity. The electronic device records each person's responses as a text file that does not have the subject's gender in the filename. These files have all been placed in the same folder on your company's network. The task is given to separate all of the files into two separate folders, one for each gender, while retaining the files original format. Unfortunately, the gender is located only within each file. Solutions do exist without the X statement. These tend to involve some variants of the following four steps: reading all of the files, determining their gender, rewriting each file to

the correct folder, and subsequent file deletion. This procedure can be made much simpler with the X statement. All that needs be done prior to invoking the X statement is to use the data step to determine if each file corresponds to a female or male and then store this information in a macro variable. Then one issues X statements conditional on the value of the macro variable to determine where to move the files.

Considering the previous scenario, the more simple file management question of file deletion may come up. Suppose after the file separation above, it was found that the coworker conducting the survey above inadvertently gave all males with subject number less than "0010", the wrong product to consider and you wish to delete these files. Again, using the X statement, these files can be deleted from within SAS.

From these scenarios, it should be obvious why the X statement is a powerful and useful tool given with Base SAS. Additionally, it should be clear that the X statement, like the Base SAS package in which it resides, is something that generally transcends occupations and programming skill levels. That is, the X statement is something that can be useful to anyone who uses SAS software.

CODE EXAMPLES

Examples of how the X statement is used will now be discussed. The syntax of the command itself is simple; one issues the statement, X '*command*', where *command* is a command from the host operating system, as the SAS Documentation states. However, this is difficult to get a feel for without seeing some examples.

EXAMPLE 1

The following is a very elementary use of the X statement and therefore is an ideal first example. A file called "Delete Me.txt" has been created in the directory H:\Files\presentation\SUGI 31\. The objective of the following statement is to delete this file.

```
x "cd H:\Files\presentation\SUGI 31\";
x "del Delete_Me.txt";
```

The first statement above changes your active directory to H:\Files\presentation\SUGI 31\. If you are familiar with DOS, you will notice that this statement would not make this the active directory in DOS unless you were already on the H:\ drive. However, regardless of what your default drive is, such a statement will work in SAS. Additionally, it has the side effect of changing the folder that is current in your SAS session. The second statement above is a DOS command to delete the file Delete_Me.txt. Depending on a SAS system option that will be discussed later, the Command Prompt (DOS Window) may remain open or may close after this invocation.

EXAMPLE 2

Consider the following statement:

```
x "%bquote(&part1) START -cwd D:\Foldername -input D:\Foldername\batchgraph.ssc -
output D:\Foldername\batchgraph.txt -logfile D:\Foldername\batchgraph.log -
quitonerror";
```

This command instructs the operating system to execute a program written in another software package. In general, this SAS code poses a solution to the second scenario listed above. However, this particular case was slightly different. An existing program written in a different software package was desirable for creation of a large set of graphs. However, the particular software package was limited in the number of graphs it could produce before having out of memory errors. It is unknown if these errors were due to the particular code written in that package or were internal to the package. In either case, a workaround to save memory was not clear after several hours of troubleshooting and reprogramming. Given the size of the batch of graphs required and the aforementioned memory error, to create these graphs using only the other software package would require over 50 iterations of starting this other software package, altering the code to state which graphs to run, running that code and then closing the software package down. Consequently, the code above, coupled with some macro code was written to avoid doing this. In the code above, the macro variable part1 (protected by a quoting function) is a path and program that is being used to execute a particular piece of code written in the other software, here batchgraph.ssc. One will note that this statement is the same type of statement one can issue in from the DOS Window on the Start Menu in Windows.

Along with the X statement, writing SAS code to pass parameters into the other software package is necessary. Here, to solve the problem, a %do loop in a SAS macro was written to repeat the following two steps: (1) create a file containing graph iteration numbers, (2) call the other software's program for those iterations. The SAS program therefore was conditionally executing the other program over the 50 iterations. By doing this, memory problems were avoided. Specifically, the SAS macro counts iterations of graph production and ensures that at any given time, the number of iterations requested of the other software package does not exceed the memory limitations (1000 graphs at a time was found to be safe). Specifically, the SAS program creates a SAS dataset with values for a counter. The

following code is embedded in a macro that essentially serves as a counter.

```
data dd1.bat;
col1=%sysevalf((1000*(&i-1))+1);
col2=%sysevalf(&i*1000);
output;
run;
```

The X statement is then given as stated above. The other software reads in the dataset, as it is capable of reading SAS datasets (note that text file could be created to pass this file to something that isn't "SAS friendly"), and processes the particular set of graphs requested. The SAS macro then loops and the process is repeated. The whole batch of graphs can then be created in the software desired by running one SAS program.

EXAMPLE 3

Consider the following statements:

```
dm 'x "cd D:\";';
dm 'x "thunder";';
dm 'x "exit";';
```

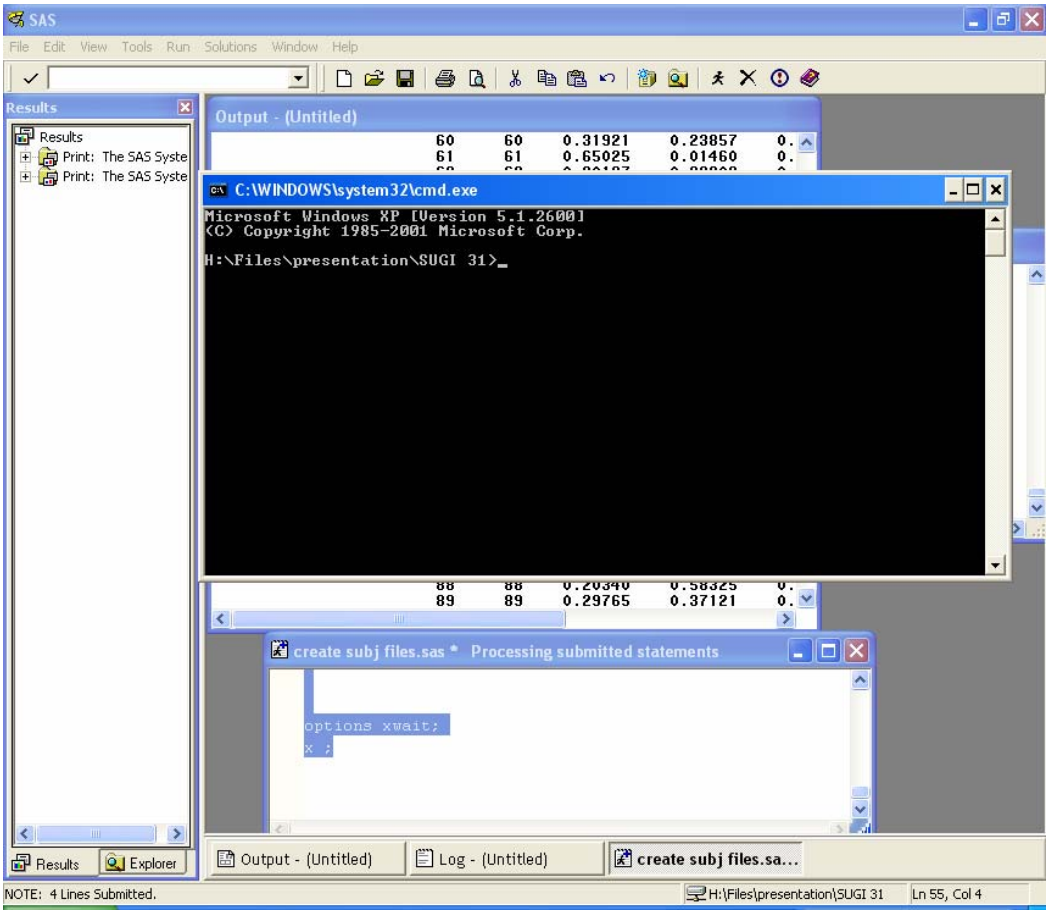
In these statements, the dm statement is being used to "submit SAS display manager or text editor commands as SAS statements." In this case, the display manager command being submitted as a SAS statement is the X command. In retrospect, the code could have been written without using the dm statement and X command, but instead with just the X statement. However, this is an interesting example because it calls attention to the dm statement. The dm statement is used to perform SAS Display Manager command line type tasks while the SAS program itself is executing. For instance, the dm statement can be used to clear the log window during execution. (SAS Institute, 2005).

```
dm log 'clear' output;
```

This code clears the log window and then makes the output window the active window.

Looking back at the three dm statements issued, this code uses the X command to change the current OS working directory to D:\, then uses the X command to call a program named thunder that resides in D:\. Finally, it exits the DOS Window. The thunder program was a compiled Visual Basic program written by this author to shade cells in Word tables consistent with values that were contained in each cell. A summary of this program can be found in the proceedings of the MWSUG 14 conference. This solution is similar, although it had a very different objective, to the solution of the first scenario listed above.

When you issue an X statement (or X command) in the Windows environment, it opens the DOS Window and submits your command. However, depending on instance, you may want the window to remain open, so you can interactively use DOS commands, or you may wish for it to close and for your program to continue executing as expected. The first of these is the default in SAS. When you use the option for the window to remain open, you may issue DOS commands as seen fit and then you must type "exit" to leave the prompt and continue executing your SAS program. The following screen capture illustrates the DOS Window as it opens in your SAS session.



However, it is often desirable to write the code and let it run without user intervention. That is, the DOS command in the X statement should be executed and SAS should immediately continue running. To close this window programmatically, you may issue the options noxwait statement in your program. You will note that in the first X statement code example listed above (the one involving the graphs), there is no X "exit" statement. This is because the noxwait system option is used. The noxwait system option "specifies that the command processor automatically returns to the SAS session after the specified command is executed. You do not have to type EXIT." (SAS Institute, 2005). In order to keep the DOS Window open, you issue the converse of the noxwait system option, which is the xwait system option. The noxwait option also applies to the X command, %sysexec and Call System. The important thing for the purposes of the X statement is that if you use the noxwait option, then the DOS Window closes and SAS continues executing as normally expected. However, it is noted that the X statement may be used solely to change a SAS current folder, such as the following statement,

```
x "D:"
```

In this instance, regardless of the xwait option, the DOS Window will not remain open.

In the documentation prior to SAS9, the noxwait option is not described. Hence, in the second code example, which was not written in SAS9, the noxwait system option was not available (or at least not documented), and hence, the X command is used to issue a DOS exit statement. This is one possible workaround for earlier versions of SAS if you wish the SAS program to continue executing after the X statement is submitted. This code was written in SAS Version 8.2 and on an older version of Windows. These commands may need modification to work or may not work at all with other versions of SAS or newer versions of Windows. In Version 8, the noxwait option is given for the systask command, which also executes host commands. This may be another workaround in earlier versions, although this author has not investigated this. It is recommended for SAS9 that you use the noxwait option if you wish to close the DOS Window. The options noxwait system option statement is given as follows:

```
options noxwait;
```

EXAMPLE 4

Consider now the scenario where you have the large number of files that you wish to sort into two folders, female and male. For this example, these files were randomly generated using SAS. Following is the code used to create these files.

```
data one;
do i=1 to 100;
x=ranuni(0);
y=ranuni(2);
z=ranuni(5);
if x<0.5 then gender="M";
else gender="F";
output;
end;
run;


%macro one;
%do i=1 %to 100;
data _null_; set one;
file "H:\Files\presentation\SUGI 31\subject&i..txt";
if _n_=&i;
put x y gender z;
run;
%end;
%mend one;
%one;
```

Each file is called subject#.txt and contains a line of data such as what follows:
0.1486685575 0.6780255361 M 0.7711814012

As mentioned previously, the process of sorting these involves reading in the information, and then moving the file into an appropriate folder, "Female" or "Male," based on the information contained therein. The X statement can be used to create these folders. The following code does so.

```
options noxwait;
x "cd H:\Files\presentation\SUGI 31";
x "mkdir male";
x "mkdir female";
```

After the female and male folders are created, the following macro is used to read in each file, determine the gender of the subject that the file corresponds to and store that value in a macro variable called var&i, where i is a macro variable representing the subject number. Then, the X statement is used to move the file into the female or male folder as appropriate.

```
%macro two;
%do i=1 %to 100;
data _null_;
infile "H:\files\presentation\SUGI 31\subject&i..txt";
input x y gender $ z;
call symput ("var&i",gender);
run;
%if &&var&i=F %then %do;
x "cd H:\Files\presentation\SUGI 31";
x "move subject&i..txt female";
%end;
%else %if &&var&i=M %then %do;
x "cd H:\Files\presentation\SUGI 31";
x "move subject&i..txt male";
%end;
%end;

%mend two;
%two
```

CONCLUSION

The X statement is powerful statement that comes with Base SAS and allows you to issue OS commands as part of a SAS program or interactively during a SAS session. This gives you the power to perform many actions from within a SAS session. In addition, it allows stand-alone programs to be written in SAS that do a number of tasks involving multiple programs, while having a simple application for the end user. The noxwait option is particularly useful in this sense, as it allows the DOS Window to close upon completion of the DOS command. The dm statement allows you to issue SAS commands in open code. This can be useful for such things as clearing the log during a program. In addition, it could be used to issue the X command, which does the same thing as the X statement, except it is issued from the SAS Display Manager command line.

REFERENCES

SAS Institute Inc. (1990) *SAS Language*. Cary, NC: SAS Institute, Inc.

SAS Institute Inc. 2005. **SAS OnlineDoc® 9.1.3**. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

Acknowledgements go to George R. Jerdack, The Procter and Gamble Company, for proofreading an early draft of this paper and especially to Kenton D. Juhlin, The Procter and Gamble Company, for both proofreading an early draft of this paper and for specific DOS help.

CONTACT INFORMATION

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

George J. Hurley
The Procter and Gamble Co.
Miami Valley Innovation Center, Box 30
11810 E Miami River Rd.
Cincinnati, OH 45252
Work Phone: 513-627-0692
Fax: 513-945-3441
E-mail: hurley.gj@pg.com