Paper RF-08-2015

ESSENTIALS OF MACRO QUOTING FUNCTIONS IN SAS®

Kaushal Raj Chaudhary, Sanford Research, Sioux Falls, SD.

ABSTRACT

SAS® macro language is a text processing facility. Everything including numeric values is treated as text in SAS macro language. It is also composed of special characters, such as, (comma), ; (semi colon), + (plus) and others. When a macro code contains, for example, ; (semi colon) sign, macro processor sees it as the end of SAS or macro statement rather than text. This might yield unintended result when you do not mean to end the statement. Macro quoting functions come to rescue from this situation by treating it as text. This paper will introduce macro quoting functions with examples.

INTRODUCTION

SAS macro quoting functions mask the special meaning of characters or combination of characters preventing unintended interpretation by macro processor. Thus macro quoting helps to write more robust macro code despite adding little complexity to the program.

Macro quoting functions mask the following SAS tokens.

- **Special characters:** + - * ** / < > = ^ ~ ; , # blank
  + - * ** / (Arithmetic operators)
  < > = (Comparison operators)
  ; (End of SAS Statement)
- **Mnemonic**: AND OR NOT EQ NE LE LT GE GT IN (Comparison operators)
- **Unmatched characters:** ‘ “ ” ( )
- **Macro triggers:** & %
  & (Macro variable resolution)
  % (Macro invocation)

Macro quoting functions are of two types.

- Compile time macro quoting functions
- Execution time macro quoting functions

Sometimes it might be hard to decide whether to use compile time or execution time. Russ Tyndall from SAS technical support suggests a general rule of thumb to resolve this dilemma in SAS User’s blog.

“If you can see the problem, it is a compile issue; otherwise, it is execution time.”

COMPILE TIME MACRO QUOTING FUNCTIONS

Compile time macro quoting functions mask the character string during the compilation of macro or macro statement. Compilation time essentially refers to situations where we have included special characters or symbols in our code. If the compiler sees them, it will use their conventional meaning in macro language. We don’t let this happen by using compile time quoting functions. %STR and %NRSTR (NR stands for “No Rescan “or “Not Resolved”) are the compile time macro quoting functions.

Syntax:

```
%STR (character string)
%NRSTR (character string)
```

%STR and %NRSTR

%STR is the most commonly used compile time macro quoting function. It masks all special characters except the macro triggers '&' and '%'. This function is also used to preserve leading and trailing blanks. %NRSTR is similar to %STR and it will also masks macro symbols (& and %), thus preventing resolution of macro variable or macro. One limitation of %STR and %NRSTR is that they can't quote unmatched characters directly. If the argument to %STR or %NRSTR contains an unmatched single or double quotation mark or an unmatched parenthesis, a % sign should precede each of these characters.

In example below we are trying to create a macro variable "mvar" with a value of "proc print data=sashelp.calss; run". If we don't use %STR, the first semi colon (;) in the value of macro variable marks the end of %LET statement producing unintended result.

```
%let mvar=proc print data=sashelp.class; run;;
%put &mvar;
```

Log output:
Proc print data=sashelp.class

Using %STR solves the problem.

```
%let mvar=%str(proc print data=sashelp.class; run;);
%put &mvar;
```

Log output:
proc print data=sashelp.class; run;

In the program below value of macro variable "myvar" contains unmatched double quotation mark. % sign has been used before it to get around the problem.

```
%let myvar=%str(%"Philip);
%put &myvar;
```

Log output:
"Philip

The following example illustrates the use of %NRSTR. If we want to create macro variable called "name" with a value of Alex&Philip, %NRSTR should be used instead of %STR since macro variable value includes macro symbol &. %NRSTR will mask &.

```
%Let name=%nrstr(Alex&Philip);
%put &=name;
```

Log output:
name= Alex&Philip

EXECUTION TIME MACRO QUOTING FUNCTIONS

Execution time macro quoting functions mask the character string during execution of macro or macro statement. During macro variable resolution or macro execution, we have no way to know what they would resolve to. They might contain characters that have meaning to macro language. To handle these situations execution time macro quoting functions are used. %BQUOTE, %NRBQUOTE, and %SUPERQ are the execution time macro quoting functions

Syntax:
%BQUOTE (character string| text expression)
%NRBQUOTE (character string| text expression)
%SUPERQ (name of macro variable)

%BQUOTE and %NRBQUOTE

%BQUOTE also called “Blind Quote” and %NRBQUOTE are another execution time macro quoting function. They mask the resolved value from macro variable resolution or macro execution. %NRBQUOTE also masks macro triggers (&, %). They continue to resolve any macro variable references as far as possible before starting quoting. If a macro call cannot resolve, it issues a warning and masks the resolved value.

```
data _null_;
  call symputx('macvar','Alex&Philip');
run;
%put %bquote(&Macvar);
%put %nrbquote(&Macvar);
```

Log Output:
```
WARNING: Apparent symbolic reference PHILIP not resolved.
WARNING: Apparent symbolic reference PHILIP not resolved.
14   %put %bquote(&Macvar);
Alex&Philip
15   %put %nrbquote(&Macvar);
WARNING: Apparent symbolic reference PHILIP not resolved.
Alex&Philip
```

In the program above %BQUOTE and %NRBQUOTE mask the resolved value of macro variable ‘macvar’. Since the resolved value contains macro variable reference (&Philip), they try to resolve it issuing a warning.

%SUPERQ

%SUPERQ is another powerful macro quoting function. The argument of superq is the name of macro variable without ampersand (&) or a text expression that resolve to macro variable name. %SUPERQ does not attempt any resolution of its argument while %NRBQUOTE tries to resolve. Because of this, %SUPERQ does not issue any warning if resolved value contains any macro variable reference. **So it is recommended to use %SUPERQ instead of %NRBQUOTE**. An example is illustrated below.

```
data _null_;
  call symputx('Macvar','Alex&Philip');
run;
%put %nrbquote(&Macvar);
%put %superq(Macvar);
```

Log output:
```
WARNING: Apparent symbolic reference PHILIP not resolved.
19   %put %nrbquote(&Macvar);
Alex&Philip
20   %put %superq(Macvar);
Alex&Philip
```

%SUPERQ does not generate any warning.

QUOTING CHARACTER FUNCTIONS

Some character or text macro functions have their quoting counterpart starting with letter “Q”, such as %QSUBSTR and %QSCAN. They mask special character and mnemonic operators. In example below %QSUBSTR returns the substring of resolved value of macro variable “name” and masks &.

```
%let name=%nrstr(Alex&Philip);
%let macvar=%qsubstr(&name,1,5);
%put &macvar;
```

Log output:
Alex&

%UNQUOTE

Quoting remains effective until it is removed. %UNQUOTE helps restore the meaning of any special characters and mnemonics masked by quoting function during macro execution.

Syntax:
%UNQUOTE (Character string| text expression)

Here macro variable "Conf" will not resolve as it is quoted using %NRSTR. %UNQUOTE reestablishes the usual meaning of "&" in macro language and "Conf" will resolve to its value.

```
%let Conf = MWSUG2015;
%let macvar = %unquote(%nrstr(&Conf));
%put &macvar;
```

Log output:
MWSUG2015

CONCLUSION

This paper introduced macro quoting functions in SAS with lucid examples. Although it is one of the difficult topics in SAS macro language, I hope this paper provided some basic knowledge of macro quoting functions.

REFERENCES

Whitlock, Ian "A Serious Look at Macro Quoting" http://www2.sas.com/proceedings/sugi28/011-28.pdf

Dunn, Toby "Macro Quoting" http://analytics.ncsu.edu/sesug/2008/CS-049.pdf

Carpenter, Art. 2003. "Macro Quoting Functions, Other Special Character Masking Tools, and How to Use Them." Proceedings of the Twelfth Annual Northeast SAS User Group Conference. Washington, DC.

SAS® Institute Inc. 2014. *Macro Functions*. SAS® 9.2 Macro Language: Reference. Cary, NC: SAS® Institute Inc.

Tyndall, Russ "Macro quoting made easy" http://blogs.sas.com/content/sgf/2014/08/15/macro-quoting-made-easy/

Contact Information

Your comments, questions, and suggestions are valued and encouraged. Contact the authors at:

Kaushal Raj Chaudhary
Sanford Research
Sioux Falls, SD
Email: kaushal.chaudhary@SanfordHealth.org