

ABSTRACT

In a world where technological advances allow tasks to be carried out at a rapid pace, are there any hidden gems within SAS that could save us more time if we only knew they were there? The word "macro" can make some programmers uneasy, but SAS holds a wealth of automatic system-defined macros that are simple to use as functions and can help reduce both the development time and run time of programs.

The paper supporting this poster provides further details on how the macro processor works when executing code and provides examples of SAS quoting macro functions and macro functions that can evaluate expressions, highlighting the situations they can be used in, the options needed and possible restrictions.

Key Features of the Macro Language

The macro language is a string based language and carries out tasks by interacting with the macro processor. There are two delimiters used within the macro language:

- % = macro
- & = macro variable

There are special characters used within the macro language that often need to be displayed as text, this can cause confusion unless the meaning of these special characters is masked – quoting macros are able to do this. An example of the type of special characters that can be masked are:

```
blank ) = LT ; ( |
GE ~ + AND GT ^ —
OR IN ~ * NOT % ,(comma)
/ EQ & ' < NE #
" > LE
```

System Macros

Situation

- Does a text string or variable have unmatched " or (or %?

Solution

- Use **%BQUOTE** as this doesn't require an additional % to mask unmatched special characters.

Example

- %let text="Investigator's Decision";
- %put %bquote(&text);
- Result = Investigator's Decision

Situation

- Is it necessary to substring text that contains &?

Solution

- Use **%QSUBSTR** as this will mask the special character so the macro variable will not be resolved but will quote the characters as they are.

Example

- If &a=AE, &b=MH and &c=&a&b
- %put %qsubstr(&c, 1, 2)
- Result=&a not "AE"

Situation

- Is it necessary to evaluate an arithmetic expression that contains a decimal place?

Solution

- Use **%SYSEVALF** as this will use floating point arithmetic and will return a value that has been formatted using BEST32. format.

Example

- %let y=%sysevalf(20 + 2.534);
- %put &y
- Result=22.534

%SUPERQ Macro

There are a number of different quoting macros available in SAS however some of these macros have restrictions, such as certain special characters that will not be masked correctly or log errors produced that should be removed where possible.

The %BQUOTE macro cannot successfully mask a single & however this can be achieved by using %SUPERQ instead of %BQUOTE. The list below shows some of the key features of the %SUPERQ macro:

%SUPERQ can mask all special characters and will quote the value of the macro variable without allowing any resolution to occur.

Since the macro variable is not resolved %SUPERQ does not result in any warning messages that a macro variable reference or macro invocation is not resolved.

The argument for %SUPERQ should be the macro variable name without &
%let text="Investigator's Decision";
%superq(text);