ABSTRACT

For years, macro programmers have wanted a function that has been available to DATA Step programmers: the equivalent of a PUT _ALL_ statement for macro variables. Version 7 promises many things including the ability to dump the macro symbol tables. But, for the many users who cannot wait for Version 7, there is a way to accomplish much of the same functionality using little known consequences of features currently in Version 6. This paper will show how to dump the macro symbol tables using Version 6.08 of the SAS® language.

INTRODUCTION

Before exploring ways to display the macro symbol tables, consider how the SAS System stores macro variables. User-defined macro variables are stored in either Global or Local symbol tables. The Global symbol table contains user-defined macro variables that are available throughout the SAS session. Local symbol tables contain the variables of last macro, while it is executing.

Normally, there is only one Local symbol table. But, when nested macros are encountered, multiple Local symbol tables are created. A Local symbol table exists for each level of nesting. For example, if macro A1 contains macro B1, there will be a local table for A1, and B1. While executing macro A1, the Local symbol table for A1 and the Global symbol table are available. While executing macro B1 from within A1, the Local symbol tables for A1 and B1 and the Global symbol table are available, and so on.

THE NAMES OF USER-DEFINED MACRO VARIABLES

Often, finding out the names of user-defined macro variables can be difficult, especially when macros generate names (e.g., & &A1i). Each macro can be examined, and names written down. Or, a little known 'feature' (setting the system option MSYMTABMAX to 0) can be used to cause the SAS System to store the symbol tables to disk.

When symbol tables are stored on disk, they appear as CATALOGs of the SASWORK library. The name of each table has the prefix SASST. The Global symbol table is named SASSTO. The first Local symbol table is named SASST1. The next local table (if there is a nested macro) is named SASST2, and so on.

You can actually use the Catalog Window to view the names of macro variables. But you cannot use the Catalog Window to view the values of macro variables.

Debugging tools such the SYMBOLGEN option can be used to display the resolved value of each macro variable as the macros are invoked. However, if you want to view only selected macro variables, you will need to employ the %PUT statement.

USING %PUT TO DISPLAY MACRO VARIABLES

The SAS System provides the %PUT statement that can be used similarly to the PUT statement of the DATA step. While the PUT statement with the _ALL_ option will automatically display all variables of the DATA step, the %PUT statement has no documented option to automatically display all available macro variables. Instead, %PUT statements must contain explicitly references to the name of each macro variable that is to be displayed.

Also, while global macro variables can be displayed at almost any time during the SAS session, local macro variable can be displayed only while 'inside' the macro that created the variable. Attempts to display local macro variables 'outside' of the creating macro(s) can caused '...unresolved reference...' error messages.

But the SAS System provides still another way to display the value of macro variables.

USING SAS/AF® TO DISPLAY MACRO VARIABLES

SAS/AF is one of the most powerful features of the SAS Language (the MACRO Facility is another). Like a DATA Step program, a SAS/AF program can extract the value of a macro variable (using the SYMGET statement). But unlike a DATA Step program, SAS/AF programs can operate at a different levels and can be invoked at any time during a SAS session. Using a DM statement, an AF command can be placed even inside a macro.

Also, the Screen Control Language SCL used in SAS/AF programs include some programming features that allow the display of the member names of any CATALOG entry. An SCL program can be written that will:

a. extract a list of symbol tables,

b. extract the names of all macro variables of each symbol table,

c. display the name of the table, each macro variable name and value.

CONCLUSION

Using the system option MSYMTABMAX and a SAS/AF SCL program you can display the names and the values of macro variables stored in the macro symbol tables. Remember, macro variables that are unavailable, appear in a SCL program (via SYMGET) as though it has a blank value.

Unfortunately, this paper only deals with user-defined macro variables. Automatic macro variables (e.g., SYSJOBID) can not be displayed using the functions described above or below.

EXAMPLE:

The following can be used to display macro variable names and
values from within SAS macros. Note, in the accompanying example, a special message is printed when the macro variable value is blank. When a macro variable value is unavailable, the SCL SYMGET function provides a blank value:

Test Macro Code:

```sas
options msymtabmax = 0;
%let x = 1;
%let z = 0;
%put before macro test1;
%macro test1;
%let x1 = 2;
%macro test11;
%let x11 = 3;
%mend;
%test11;
%put before AF statement;
%macro test1;
%let x = 2;
%put before macro test1;
%mend;
%test1;
%put before AF statement;
%mend;
```

The 'screen' for 'test.program' looks like (the screen will not be used):

```
& &
```

The SCL code for 'test.program' looks like:

```sas
length level name $8 line $74;
init:
  control enter;
  re = filename('derek','"&saswork1"');
  fid = fopen('derek5', 'c:\derek16.txt');
  rc = fseek(fid, 0);
  rc = fclose(fid);
  rc = rc;
  did = dopen('derek');
  num = dnum(did);
  call execmd('msg on;end');
return;
main:
do i = 1 to num;
  filename = scan(dread(did,i),1);
  if filename =: 'SASST' then do;
    put filename =;
    call execmd('build work.');[trim(filename)]
    end;
  else;
    level = scan(filename, '.', 1);
    name = scan(filename, 1, '.');
    value = trim(symget(name));
    if value = ' ' then
      substr(line,36) = ' ** blank or may be unavailable **';
    else
      substr(line,36) = value;
  end;
end;
rc = dclose(did);
return;
Dump of Macro Symbol Tables
```

Output after submitting the macro:

```
1 options msymtabmax = 0;
2 %let x = 1;
3 %let z = 0;
4 %put before macro test1;
before macro test1
5 %macro test1;
6 %let x1 = 2;
7 %put before macro test11;
8 %macro test11;
9 %let x11 = 3;
%mend;
%test11;
%put after AF statement;
%mend;
```

SAS, SAS/AF are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.