Paper 80-28

Return Code From Macro; Passing Parameter By Reference

Hsiwei Yu (Michael), People Technology Solutions Inc, Edison, NJ
Gary Huang, InfoStat Inc., Belle Mead, NJ

ABSTRACT

For software reusability, a macro (function or utility) must clearly inform the caller about its execution status so as to make itself easily usable by other programmers. A macro function which only writes error messages to SAS log is not user-friendly in terms of passing its result back to the caller. Here we propose writing macro routine to return its execution status via macro variable, i.e. passing parameter by reference. Other programmers using such macro can easily determine its result and then proceed accordingly. This new way of passing results makes a macro convenient for other people to use, thus promoting software reusability.

INTRODUCTION

A macro routine is like a black box. Given valid inputs from a caller, it shall perform the desired actions. It is critical for the external user of this routine be able to discern what had happened. For a real life example, when I deposit money into a bank, I certainly would not feel comfortable without getting a receipt back from bank teller.

Sometimes a macro routine designer would write out warning or error messages to the SAS log exclusively, in the hope that the caller, i.e. user, would review the SAS log. This old-fashioned way is unacceptable because the caller must *read* the SAS log to find out. Not an impossible task, but unnecessary burden on the caller, thus preventing software reusability.

Information is exchanged between the caller of a macro routine and the macro routine. Most people are familiar with the caller passing parameters into a routine. This is data flow from caller to the called routine. However, what is not emphasized enough is that data *must* also flow from the called routine back to the caller. This transfer of knowledge from the called macro routine to the caller must be programmed into macro routine, not as an after-thought, but an integral part of good programming practice.

MESSAGE BURIED IN SAS LOG

Consider a macro writing a message to a fixed location, be it SAS log or external file, like:

```
%macro doTask( inType= );
…
%if &inType = 1 %then %do;
/* Do something then write out a message */

data _null_;
file out;
 put 'Your input is …'
   / 'Execution status is great!';
run;
%end;
%else %if &inTYpe = 2 %then %do;
..
%end;
%mend doTask;
```

The deficiency is that this macro does NOT pass its execution result back to the caller. Anyone wishing to use this macro is never sure what happened. This defect limits the usefulness of a macro routine and hinders software reuse. Can a macro routine send its execution result back to the caller *in plain sight*?

PARAMETER BY VALUE AS INPUT

For a macro definition like below:

```
%macro doTask( inType= );
%if &inType = 1 %then %do;
…
%else %if &inType = 2 %then %do;
…
%mend  doTask;

%doTask( inType= 2 );
```

Think of the "inType" as input parameter, because the caller gives it a value for the macro routine to use. Can we have another type of parameter for returning result back to the caller?

PARAMETER BY REFERENCE FOR RETURNING RESULT

If a parameter's value is the *name* of a global macro variable, then inside the called macro routine this global macro variable is accessible and can be assigned value. Once the called routine is finished, the global macro variable still retains its new value. As such, the caller can check and determine the message being passed back from the called routine. Here is a sample:

```
/* ***
The value of returnCode must be the name of a
global macro variable.
As such, returnCode is a parameter for
returning information back to the caller.
*** */

%macro doTask( inType=, returnCode= );
%if &inType = 1 %then %do;
…
/* ***
Note &returnCode is assigned value.
Because returnCode's value is the name of a
global macro variable,
&returnCode resolves to the global macro
variable.
*** */
   %let &returnCode= 0;

%end;
%else %if &inType = 2 %then %do;
…
   %let &returnCode= 5;
%end;

%mend doTask;
```

We say a parameter is passed by reference if the parameter's value is the *name* of a global macro variable, and such parameter is called a reference parameter. Passing parameter by reference makes it possible for the called routine to send information back to the caller cleanly and efficiently.

Note the strange syntax of assigning value to the reference parameter inside the called macro, like:

```
   %let &returnCode= 0;
```

This is necessary because the value of this reference parameter, returnCode, is the *name* of a global macro variable. We are directly assigning value to the global macro variable represented by the reference parameter, &returnCode.

Below is a sketch for other programmer to use the return information from %doTask:

```
%macro programUsingDoTask;
%global RC;
%let RC=;
%doTask(inType= 1, returnCode= RC);

/* Now caller can decide what to do next! */
%if &RC = 0 %then %do;
/* ***
Since it's a success, do more things!
*** */
…

%end;

%else %do;
/* Something bad happened. */
…

%end;

%mend programUsingDoTask;
```

SASHELP.VMACRO TO CHECK EXISTENCE

Since the reference parameter's value must be an existing global macro variable, the called routine should verify its existence before assigning value to it. The SASHELP.VMACRO view can be used to check for global macro variable's existence, like:

```
%macro doTask( inMsg=, returnCode= );
/* ***
If a macro variable exists, it can be found
in SASHELP.VMACRO view.
*** */
data existingMacroVars;
 set sashelp.vmacro( where=
( upcase( name ) = %upcase( "&returnCode" )
and scope = 'GLOBAL' ));
run;

%let dsid=%sysfunc(open(existingMacroVars ));
%let found=%sysfunc(attrn( &dsid, nobs ));
%let dsid=%sysfunc(close( &dsid ));

%if &inType = 1 %then %do;
…
   /* Assign value to the global macro
variable, &returnCode, only if it does exist!
*/
   %if &found gt 0 %then
   %let &returnCode= 0;

%end;
%else %if &inType = 2 %then %do;
…
   %if &found gt 0 %then
   %let & returnCode= 5;


%end;

%mend  toTask;
```

CONCLUSION

The concept of returning execution result from function or routine back to the caller is critical and widely accepted in many programming environments, such as C and C++. Within the confine of SAS Macro Language, the same is achieved by passing the *name* of a global macro variable into a routine, and the macro routine directly assigns value to the reference parameter. This way, the caller will receive information about the called routine's status or anything the called routine deem important for the caller to know.

Documentation of a macro utility must clearly inform its reader if a parameter is value or reference parameter, for example:

```
/* ***
inType is a input value only parameter.

returCode is a reference parameter for its
value must be the name of a global macro
variable.

The purpose of %doTask is to …
*** */
%macro doTask( inType=, returnCode= );
```

Macro programs will become more complex after adopting this convention between the caller and the called routine, however this is necessary in order to promote software reusability. A macro routine so designed and implemented is much more useful to other programmers, because they know for sure the success or failure in a called routine and can take corrective actions as appropriate.

CONTACT INFORMATION

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

Hsiwei Yu (Michael)
People Technology Solution Inc.
60 Stephenville Pkwy
Edison, NJ 08820
Work Phone: 908-740-2494
Email: hsiwei_yu@yahoo.com

Gary Huang
InfoStat Inc.
50 Green Ave.
Belle Mead, NJ 08502
Work Phone: 908-740-7571
Emai: Gary0nline@yahoo.com