**PhUSE 2010**

**Paper P005**

# Autocall Macros – A Quick Overview

Vinod Panangattiri Parambil, Roche Products Ltd, Welwyn Garden City, United Kingdom

## AUTOCALL MACRO

Autocall macro is a facility within the SAS system in order to use the same SAS macro code for multiple SAS programs by storing the macro in a particular location. It benefits a lot to the programmer by permitting faster updates and better consistency across all programs.

## INTRODUCTION

By default a set of autocall macros are included in a library, called "Autocall Library" within the SAS system supplied by SAS Institute. In order to make use of autocall macros we must include the autocall library in the list of search locations. This paper will provide details regarding some of the SAS defined autocall macros and how these macros can be used within the SAS system.

## HOW TO USE

In order to access the autocall macro within a SAS program, the following two SAS options are required:

MAUTOSOURCE - tells SAS to activate the autocall facility
SASAUTOS - tells SAS where to look for the macros.

### MAUTOSOURCE:

MAUTOSOURCE option will be the default setting which specifies that the autocall facility is available within the SAS session.

```
options mautosource;
```

NOMAUTOSOURCE option specifies that the autocall facility is not available.

```
options nomautosource;
```

### SASAUTOS:

```
options sasautos = Library #1;

options sasautos = (Library #1, Library #2, . . . , Library #n);
```

where `Library #1, . . ., Library #n` are the references to the source libraries which contain the autocall macros.

The following section discusses some of the autocall macros and their use:

### (i) %CMPRES

%CMPRES macro returns the argument passed to it in an unquoted form with multiple blanks compressed to single blanks and with leading and trailing blanks removed.

**Syntax**:

```
%cmpres(argument);
```

**Example:**

```
proc sql noprint;
  select count(type) into : num
    from sashelp.vcolumn
    where libname = "SASHELP" and memname = "CLASS" and type = "num";

  select count(type) into : char
    from sashelp.vcolumn
    where libname = "SASHELP" and memname = "CLASS" and type = "char";
quit;

%let x = %str(SASHELP.CLASS contains &num numeric and &char character variables);
%let y = %cmpres(&x);

%put ***x = &x***;
%put ***y = &y***;
```

The following lines are written to the Log File:

```
***x = SASHELP.CLASS contains        3 numeric and        2 character variables***
***y = SASHELP.CLASS contains 3 numeric and 2 character variables***
```

**(ii) %LEFT**

%LEFT macro returns the argument passed to it without any leading blanks in an unquoted form. The syntax for its use is similar to that of native macro functions.

**Syntax:**

```
%left(argument);
```

**Example:**

```
proc sql noprint;
  select nobs into : nobs
    from sashelp.vtable
    where libname = "SASHELP" and memname = "CLASS";
quit;

%let x = %str(SASHELP.CLASS contains &nobs observations);
%let y = %str(SASHELP.CLASS contains %left(&nobs) observations);

%put ***x = &x***;
%put ***y = &y***;
```

The following lines are written to the Log File:

```
***x = SASHELP.CLASS contains       19 observations***
***y = SASHELP.CLASS contains 19 observations***
```

**(iii) %LOWCASE**

%LOWCASE macro returns the argument passed to it unchanged except that all upper-case alphabetic characters are changed to their lower-case equivalents

**Syntax:**

```
%lowcase(argument);
```

**Example:**

```
proc sql noprint;
  select distinct (name) into : vars separated by " "
    from sashelp.vcolumn
    where libname = "SASHELP" and memname = "CLASS";
quit;

%let x = &vars;
%let y = %lowcase(&vars);

%put ***x = &x***;
%put ***y = &y***;
```

The following lines are written to the Log File:

```
***x = AGE HEIGHT NAME SEX WEIGHT***
***y = age height name sex weight***
```

**(iv) TRIM**

%TRIM macro returns the argument passed to it without any trailing blanks in an unquoted form.

**Syntax:**

```
%trim(argument);
```

**Example:**

```
data _null_;
  a = 1;
  call symput ('x', a);
run;

%put ***&x***;
%put ***%trim(&x)***;
```

The following lines are written to the Log File:

```
***           1***
***1***
```

**(v) %VERIFY**

%VERIFY macro returns the position of the first character in the argument that is not in the target value. If every character that is in the argument is also in the target value then this function returns a value of 0.

**Syntax:**

```
%verify(argument,target);
```

**Example:**

```
%let scanlist = ABC123;
%let text1 = ABC_1;
%let text2 = AC23;

%put ***%verify(&text1,&scanlist)***;
%put ***%verify(&text2,&scanlist)***;
```

The following lines are written to the Log File:

***4***
***0***

## CONCLUSION

The paper describes some of the sas defined autocall macros and how it can be used with examples.

## CONTACT INFORMATION

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

Vinod Panangattiri Parambil
Roche Products Ltd.
6 Falcon Way, Shire Park
Welwyn Garden City
AL7 1TW
United Kingdom
Work Phone: (44) 1707 36 6004
Email: vinod.panangattiri_parambil@roche.com

**APPENDIX**

The SAS code for the above specified autocall macros are given below.

**(1) %CMPRES**

```
%macro cmpres(text);
  %local i;
  %let i=%index(&text,%str(  ));
  %do %while(&i^=0);
      %let text=%qsubstr(&text,1,&i)%qleft(%qsubstr(&text,&i+1));
      %let i=%index(&text,%str(  ));
  %end;
  %left(%qtrim(&text))
%mend;
```

**(2) %LEFT**

```
%macro left(text);
  %local i;
  %if %length(&text)=0 %then %let text=%str( );
  %let i=%verify(&text,%str( ));
  %if &i %then %substr(&text,&i);
%mend;
```

**(3) %LOWCASE**

```
%macro lowcase(string);
  %sysfunc(lowcase(%nrbquote(&string)))
%mend;
```

**(4) %TRIM**

```
%macro trim(value);
  %local i;
  %do i=%length(&value) %to 1 %by -1;
    %if %qsubstr(&value,&i,1)^=%str( ) %then %goto trimmed;
  %end;
  %trimmed: %if &i>0 %then %substr(&value,1,&i);
%mend;
```

**(5) %VERIFY**

```
%macro verify(text,target);
  %local i;
  %if %length(&text)=0 | %length(&target)=0 %then %do;
      %put ERROR: ARGUMENT TO VERIFY FUNCTION MISSING.;

      %goto errout;
  %end;
  %do i=1 %to %length(&text);
      %if ^%index(&target,%qsubstr(&text,&i,1)) %then %goto verfnd;
  %end;
  %verfnd: %if &i>%length(&text) %then 0;
           %else &i;
  %errout:
%mend;
```