CHAPTER

12

Macro Language Elements

Introduction **145**
Macro Statements **145**
Statements That Perform Automatic Evaluation **147**
Macro Functions **147**
Character Functions **148**
Evaluation Functions **149**
Quoting Functions **149**
Compilation Quoting Functions **150**
Execution Quoting Functions **150**
Quotation Marks and Parentheses without a Match **151**
Other Functions **151**
Automatic Macro Variables **152**
Interfaces with the Macro Facility **153**
Selected Autocall Macros Provided with SAS Software **154**
Required System Options for Autocall Macros **155**
Using Autocall Macros **155**
Selected System Options Used in the Macro Facility **156**

Introduction

The SAS macro language consists of statements, functions, and automatic macro variables. This chapter defines and lists these elements. Also covered are the interfaces to the macro facility provided by base SAS, the SQL procedure, and Screen Control Language as well as selected autocall macros and macro system options. For details on each element, see Chapter 13, “Macro Language Dictionary.”

Macro Statements

A macro language statement instructs the macro processor to perform an operation. It consists of a string of keywords, SAS names, and special characters and operators, and it ends in a semicolon. Some macro language statements are allowed only in macro definitions, but you can use others anywhere in a SAS session or job, either inside or outside macro definitions (referred to as open code). Table 12.1 on page 146 lists macro language statements that you can use in both macro definitions and open code.

Table 12.1 Macro Language Statements Allowed in Macro Definitions and Open Code

Statement	Description
%* comment	designates comment text
%DISPLAY	displays a macro window
%GLOBAL	creates macaro variables that are available during the execution of an entire SAS session
%INPUT	supplies values to macro variables during macro execution
%KEYDEF	assigns a definition to or identifes the definition of a function key
%LET	creates a macro variable and assigns it a value
%MACRO	begins a macro definition
%PUT	writes text or the values of macro variables to the SAS log
%SYSCALL	invokes a SAS call routine
%SYSEXEC	issues operating system commands
%SYSLPUT	defines a new macro variable or modifies the value of an existing macro variable on a remote host or server
%SYSRPUT	assigns the value of a macro variable on a remote host to a macro variable on the local host
%WINDOW	defines customized windows

Table 12.2 on page 146 lists macro language statements that you can use only in macro definitions.

Table 12.2 Macro Language Statements Allowed in Macro Definitions Only

Statement	Description
%DO	begins a %DO group
%DO, Iterative	executes statements repetitively, based on the value of an index variable
%DO %UNTIL	executes statements repetively unti la condition is true
%DO %WHILE	executes statements repetitively while a condition is true
%END	ends a %DO group
%GOTO	branches macro processing to the specified label
%IF-%THEN/%ELSE	conditionally processes a portion of a macro
%label:	identifies the destination of a %GOTO statement
%LOCAL	creates macro variable that are available only during the execution of the macro where they are defined
%MEND	ends a macaro definition

Statements That Perform Automatic Evaluation

Some macro statements perform an operation based on an evaluation of an arithmetic or logical expression. They perform the evaluation by automatically calling the %EVAL function. If you get an error message about a problem with %EVAL when a macro does not use %EVAL explicitly, check for one of these statements. The macro statements that perform automatic evaluation are:

%DO *macro-variable=expression* %TO *expression* <%BY *expression*>;

%DO %UNTIL(*expression*);

%DO %WHILE(*expression*);

%IF *expression* %THEN *action*;

For details on operands and operators in expressions, see Chapter 6, "Macro Expressions."

Macro Functions

In general, a macro language function processes one or more arguments and produces a result. You can use all macro functions in both macro definitions and open code. Macro functions include character functions, evaluation functions, and quoting functions. The macro language functions are listed in Table 12.3 on page 147.

Table 12.3 Macro Functions

Function	Description
%BQUOTE, %NRBQUOTE	mask special characters and mnemonic operators in a resolved value at macro execution.
%EVAL	evaluates arithmetic and logical expressions using integer arithmetic.
%INDEX	returns the position of the first character of a string.
%LENGTH	returns the length of a string.
%QUOTE, %NRQUOTE	mask special characters and mnemonic operators in a resolved value at macro executin. Unmatched quotation marks ('") and parentheses (())must be marked with a preceding %.
%SCAN, %QSCAN	search for a warod specified by its number. %QSCAN masks special characters and mnemonic operators in its result.
%STR, %NRSTR	mask special characters and mnemonic operators in constant text at macro compilation. Unmatched quotation marks ('") and parentheses (())must be marked with a preceding %.
%SUBSTR, %QSUBSTR	produce a substring of a characater string. %QSUBSTR masks special characters and mnemonic operators in its result.
%SUPERQ	masks all special characters and mnemonic operators at macro execution but prevents resolution of the value.
%SYSEVALF	evaluates arithmetic and logical expressions using floating point arithmetic.
%SYSFUNC, %QSYSFUNC	execute SAS functions or user-written functions. %QSYSFUNC masks special charactaers and mnemonic operators in its result.

Function	Description
%SYSGET	returns the value of a specified host environment variable.
%SYSPROD	reports whether a SAS software product is licensed at the site.
%UNQUOTE	unmasks all special characters and mnemonic operators for a value.
%UPCASE, %QUPCASE	convert characters to uppercase. %QUPCASE masks special characters and mnemonic operators in its result.

Character Functions

Character functions change character strings or provide information about them. Table 12.4 on page 148 lists the macro character functions.

Table 12.4 Macro Character Functions

Function	Description
%INDEX	returns the position of the first character of a string.
%LENGTH	returns the length of a string
%SCAN, %QSCAN	search for a word that is specified by a number. %QSCAN masks special characters and mnemonic operataors in its result.
%SUBSTR, %QSUBSTR	produce a substring of a character string. %QSUBSTR masks special characters and mnemonic operators in its result.
%UPCASE, %QUPCASE	convert characters to uppercase. %QUPCASE masks special charactaers and mnemonic operators in its result.

For macro character functions that have a Q form (for example, %SCAN and %QSCAN), the two functions work alike except that the function beginning with Q masks special characters and mnemonic operators in its result. In general, use the function beginning with Q when an argument has been previously masked with a macro quoting function or when you want the result to be masked (for example, when the result may contain an unmatched quotation mark or parenthesis). For details, see Chapter 7, “Macro Quoting.”

Many macro character functions have names corresponding to SAS character functions and perform similar tasks (such as %SUBSTR and SUBSTR). But, macro functions operate before the DATA step executes. Consider this DATA step:

```
data out.%substr(&sysday,1,3);     /* macro function */
   set in.weekly (keep=name code sales);
   length location $4;
   location=substr(code,1,4);         /* SAS function */
run;
```

Running the program on Monday creates the data set name OUT.MON, as shown:

```
data out.MON;                      /* macro function */
   set in.weekly (keep=name code sales);
   length location $4;
   location=substr(code,1,4);         /* SAS function */
run;
```

Suppose that the IN.WEEKLY variable CODE contains the values cary18593 and apex19624. The SAS function SUBSTR operates during DATA step execution and assigns these values to the variable LOCATION, **`cary`** and **`apex`**.

Evaluation Functions

Evaluation functions evaluate arithmetic and logical expressions. They temporarily convert the operands in the argument to numeric values. Then, they perform the operation specified by the operand and convert the result to a character value. The macro processor uses evaluation functions to:

- make character comparisons
- evaluate logical (Boolean) expressions
- assign numeric properties to a token, such as an integer in the argument of a function.

For more information, see Chapter 6. Table 12.5 on page 149 lists the macro evaluation functions.

Table 12.5 Macro Evaluation Functions

Function	Description
%EVAL	evaluates arithmetic and logical expressions using integer arithmetic
%SYSEVALF	evaluates arithmetic and logical expressions using floating point arithmetic

%EVAL is called automatically by the macro processor to evaluate expressions in the arguments to the statements that perform evaluation, listed on “Statements That Perform Automatic Evaluation” on page 147, and in the following functions:

%QSCAN(*argument,n*<*,delimiters*>)

%QSUBSTR(*argument,position*<*,length*>)

%SCAN(*argument,n*<*,delimiters*>)

%SUBSTR(*argument,position*<*,length*>)

Quoting Functions

Macro quoting functions mask special characters and mnemonic operators so the macro processor interprets them as text instead of elements of the macro language.

Table 12.6 on page 150 lists the macro quoting functions, and also describes the special characters they mask and when they operate. (Although %QSCAN, %QSUBSTR, and %QUPCASE mask special characters and mnemonic operations in their results, they are not considered quoting functions because their purpose is to process a character value and not simply to quote a value.) For more information, see Chapter 7, “Macro Quoting.”

Table 12.6 Macro Quoting Functions

Function	Description
%BQUOTE, %NRBQUOTE	mask special characters and mnemonic operators in a resolved value at macro execution. %BQUOTE and %NRBQUOTE are the most powerful functions for masking values at execution time because they do not require that unmatched quotation marks ('") and parentheses (())are marked.
%QUOTE, %NRQUOTE	mask special charactaers and mnemoinic operators in a resolved value at macro execution. Unmatched quotation marks ('") and parentheses (())must be marked with a preceding %.
%STR, %NRSTR	mask special characters and mnemlonic operators in constant text at macro compilation. Unmatched quotation marks ('") and parentheses (())must be marked with a preceding %.
%SUPERQ	masks all special characters and mnemonic operators at macro execution but prevents resolution of the value.
%UNQUOTE	unmasks all special charactaers and mnemonic operators for a value.

Compilation Quoting Functions

%STR and %NRSTR mask special characters and mnemonic operators in values during compilation of a macro definition or a macro language statement in open code. For example, the %STR function prevents the following %LET statement from ending prematurely. It keeps the semicolon in the PROC PRINT statement from being interpreted as the semicolon for the %LET statement.

```
%let printit=%str(proc print; run;);
```

Execution Quoting Functions

%BQUOTE, %NRBQUOTE, %QUOTE, %NRQUOTE, and %SUPERQ mask special characters and mnemonic operators in values during execution of a macro or a macro language statement in open code. Except for %SUPERQ, these functions instruct the macro processor to resolve a macro expression as far as possible and mask the result, issuing warning messages for any macro variable references or macro invocations they cannot resolve. %SUPERQ protects the value of a macro variable from any attempt at further resolution.

Of the quoting functions that resolve values during execution, %BQUOTE and %NRBQUOTE are the most flexible. For example, the %BQUOTE function prevents the following %IF statement from producing an error if the macro variable STATE resolves to OR (for Oregon). Without %BQUOTE, the macro processor would interpret the abbreviation for Oregon as the logical operator OR.

```
%if %bquote(&state)=nc %then %put North Carolina Dept. of Revenue;
```

%SUPERQ fetches the value of a macro variable from the macro symbol table and masks it immediately, preventing the macro processor from attempting to resolve any part of the resolved value. For example, %SUPERQ prevents the following %LET statement from producing an error when it resolves to a value with an ampersand, like **Smith&Jones**. Without %SUPERQ, the macro processor would attempt to resolve **&Jones**.

```
%let testvar=%superq(corpname);
     /* No ampersand in argument to %superq. */
```

(%SUPERQ takes as its argument either a macro variable name without an ampersand or a text expression that yields a macro variable name.)

Quotation Marks and Parentheses without a Match

Syntax errors result if the arguments of %STR, %NRSTR, %QUOTE, and %NRQUOTE contain a quotation mark or parenthesis that does not have a match. To prevent these errors, mark these quotation marks and parentheses by preceding them with a percent sign. For example, to store the value **`345)`** in macro variable B, write

```
%let b=%str(345%));
```

If an argument of %STR, %NRSTR, %QUOTE, or %NRQUOTE contains a percent sign that precedes a quotation mark or parenthesis, use two percent signs (%%) to specify that the argument's percent sign does not mark the quotation mark or parenthesis. For example, to store the value **`TITLE "20%";`** in macro variable P, write

```
%let p=%str(TITLE "20%%";);
```

If the argument for one of these functions contains a character string with the comment symbols **`/*`** and **`-->`**, use a %STR function with each character. For example, consider the statements:

```
%let instruct=Comments can start with %str(/)%str(*).;
%put &instruct;
```

They write to the log:

```
Comments can start with /*
```

Note: Unexpected results can occur if the comment symbols are not quoted with a quoting function. △

For more information about macro quoting, see Chapter 7.

Other Functions

Three other macro functions do not fit into the earlier categories, but they provide important information. Table 12.7 on page 151 lists these functions:

Table 12.7 Macro Quoting Functions

Function	Description
%SYSFUNC, %QSYSFUNC	execute SAS language functions or user-written functions within the macro facility.
%SYSGET	returns the value of the specified host environment variable. For details, see the SAS Companion for your operating system.
%SYSPROD	reports whether a SAS software product is licensed at the site.

The %SYSFUNC and %QSYSFUNC functions make most of the functions from base SAS software and Screen Control Language available to the macro facility. Consider these examples:

```
· /* in a DATA step or SCL program */
    dsid=open("sasuser.houses","i");
```

```
· /* in the macro facility */
      %let dsid = %sysfunc(open(sasuser.houses,i));
```

For more information on each of these functions, see Chapter 13, "Macro Language Dictionary."

Automatic Macro Variables

Automatic macro variables are created by the macro processor and they supply a variety of information. They are useful in programs, for example to check the status of a condition before executing code. When you use automatic macro variables, you reference them the same way that you do macro variables that you create, for example &SYSLAST or &SYSJOBID.

CAUTION:

Do not create macro variable names that begin with SYS. The three-letter prefix SYS is reserved for use by the SAS System for automatic macro variables. For a complete list of reserved words in the macro language, see Appendix 1, *Reserved Words in the Macro Facility*. △

For example, suppose you want to include today's day and date in a FOOTNOTE statement. Write the statement to reference the automatic macro variables SYSDAY and SYSDATE9, as shown here:

```
footnote "Report for &sysday, &sysdate9";
```

If you run the program on June 15, 2001, macro variable resolution causes the SAS System to see this statement:

```
FOOTNOTE "Report for Friday, 15JUN2001";
```

All automatic variables except for SYSPBUFF are global and are created when you invoke the SAS System. Table 12.8 on page 152 lists the automatic macro variables and describes their READ and WRITE status.

Table 12.8 Automatic Macro Variables

Variable	READ/WRITE Status
SYSBUFFR	read/write
SYSCC	read/write
SYSCHARWIDTH	read only
SYSCMD	read/write
SYSDATE	read only
SYSDATE9	read only
SYSDAY	read only
SYSDEVIC	read/write
SYSDMG	read/write
SYSDSN	read/write
SYSENV	read only
SYSERR	read only

Variable	READ/WRITE Status
SYSFILRC	read/write
SYSINDEX	read only
SYSINFO	read only
SYSJOBID	read only
SYSLAST	read/write
SYSLCKRC	read/write
SYSLIBRC	read/write
SYSMENV	read only
SYSMSG	read/write
SYSPARM	read/write
SYSPBUFF	read/write
SYSPROCESSID	read only
SYSPROCESSNAME	read only
SYSRC	read/write
SYSSCP	read only
SYSSCPL	read only
SYSSITE	read only
SYSSTARTID	read only
SYSSTARTNAME	read only
SYSTIME	read only
SYSUSERID	read only
SYSVER	read only
SYSVLONG	read only

Interfaces with the Macro Facility

The DATA step, Screen Control Language, and the SQL procedure provide interfaces with the macro facility. Table 12.9 on page 153, Table 12.10 on page 154, and Table 12.11 on page 154 list the elements that interact with the SAS macro facility.

The DATA step provides elements that enable a program to interact with the macro facility *during* DATA step execution.

Table 12.9 Interfaces from the DATA Steps

Element	Description
EXECUTE routine	resolves an argument and executes the resolved value at the next step boundary
RESOLVE function	resolves the value of a text expression during DATA step execution

Element	Description
SYMGET function	returns the value of a macro variable to the DATA step during DATA step execution.
SYMPUT routine	assigns a value produced in a DATA step to a macro variable

Screen Control Language (SCL) provides two elements for using the SAS macro facility to define macros and macro variables for SCL programs.

Table 12.10 Interfaces from Screen Control Language

Element	Description
SYMGETN	returns the value of a global macro variable as a numeric value
SYMPUTN	assigns a numeric value to a global macro variable

The SQL procedure provides a feature for creating and updating macro variables with values produced by the SQL procedure.

Table 12.11 Interfaces from the SQL Procedure

Element	Description
INTO	assigns the result of a calculation or the value of a data column

For more information, see Chapter 8, "Interfaces with the Macro Facility."

Selected Autocall Macros Provided with SAS Software

SAS Institute supplies libraries of autocall macros to each SAS site. The libraries you receive depend on the SAS products licensed at your site. You can use autocall macro without having to define or include them in your programs.

When SAS is installed, the autocall libraries are included in the value of the SASAUTOS system option in the system configuration file. The autocall macros are stored as individual members, each containing a macro definition. Each member has the same name as the macro definition it contains.

Although the macros available in the autocall libraries supplied by SAS Institute are working utility programs, you can also use them as models for your own routines. In addition, you can call them in macros you write yourself.

To explore these macro definitions, browse the commented section at the beginning of each member. See the setting of SAS system option SASAUTOS, to find the location of the autocall libraries. To view the SASAUTOS value, use one of the following:

- the OPTIONS command in the SAS Display Manager System to open the OPTIONS window
- the OPTIONS procedure
- the VERBOSE system option
- the OPLIST system option.

For details on these options, refer to Chapter 16, "SAS System Options," in *SAS Language: Reference, Version 6, First Edition*.

Table 12.12 on page 155 lists selected autocall macros.

Table 12.12 Selected Autocall Macros

Macro	Description
CMPRES and QCMPRES	compress multiple blanks and remove leading and trailing blanks. QCMPRES masks the result so special characters and mnemonic operators are treated as text instead of being interpreted by the macro facility.
COMPSTOR	compiles macros and stores them in a catalog in a permanent SAS library.
DATATYP	returns the data type of a value.
LEFT and QLEFT	left-align an argument by removing leading blanks. QLEFT masks the result so special characters and mnemonic operators are treated as text instead of being interpreted by the macro facility.
SYSRC	returns a value corresponding to an error condition.
TRIM and QTRIM	trim trailing blanks. QTRIM masks the result so special characters and mnemonic operators are treated as text instead of being interpreted by the macro facility.
VERIFY	returns the position of the first character unique to an expression.

Required System Options for Autocall Macros

To use autocall macros, you must set two SAS system options:

MAUTOSOURCE
enables the autocall facility. NOMAUTOSOURCE disables the autocall facility.

SASAUTOS=*library-specification* | (*library-specification-1…, library-specification-n*)
specifies the autocall library or libraries. For more information, see the SAS companion for your operating system.

If your site has installed the autocall libraries supplied by SAS Institute and uses the standard configuration of SAS software supplied by the Institute, you need only to ensure that the SAS system option MAUTOSOURCE is in effect to begin using the autocall macros.

Using Autocall Macros

To use an autocall macro, call it in your program with the statement %*macro-name*. The macro processor searches first in the WORK library for a compiled macro definition with that name. If the macro processor does not find a compiled macro and if the MAUTOSOURCE is in effect, the macro processor searches the libraries specified by the SASAUTOS option for a member with that name. When the macro processor finds the member, it

1 compiles all of the source statements in that member, including all macro definitions

2 executes any open code (macro statements or SAS source statements not within any macro definition) in that member

3 executes the macro with the name you invoked.

After the macro is compiled, it is stored in the WORK.SASMACR catalog and is available for use in the SAS session without having to be recompiled.

You can also create your own autocall macros and store them in libraries for easy execution. For more information, see Chapter 9, “Storing and Reusing Macros.”

Selected System Options Used in the Macro Facility

Table 12.13 on page 156 lists the SAS options that apply to the macro facility.

Table 12.13 System Options Used in the Macro Facility

Option	Description
CMDMAC	controls command-style macro invocation
IMPLMAC	controls statement-style macro invocation
MACRO	controls whether the SAS macro language is available
MAUTOSOURCE	controls whether the macro autocall feature is available
MERROR	controls whether the macro processor issues a warning message when a macro-like name (%*name*) does not match a compiled macro
MFILE	determines whether MPRINT output is routed to an external file
MLOGIC	controls whether macro execution is traced for debugging
MPRINT	controls whether SAS statements generated by macro execution are traced for debugging
MRECALL	controls whether the macro processor searches the autocall libraries for a member that was not found during an earlier search
MSTORED	controls whether stored compiled macros are available
MSYMTABMAX	specifies the maximum amount of memory available to the macro variable symbol tables(s)
MVARSIZE	specifies the maximum size for in-memory macro variable values
SASAUTOS	specifies one or more autocall libraries
SASMSTORE	specifies the libref of a SAS library containing a catalolg of stored compiled SAS macros
SERROR	controls whether the macro processor issues a warning message when a macro variable reference does not match a macro variable
SYMBOLGEN	controls whether the results of resolving macro variable references are displayed for debugging
SYSPARM	controls whether the macro processor searches the autocall libraries for a member that was not found during an earlier search

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS Macro Language: Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999. 310 pages.

SAS Macro Language: Reference, Version 8

1-58025-522-1

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999