Part 1 Macro Basics

Chapter 1 Introduction

This chapter introduces you to the fundamentals of the SAS macro language, and it includes an overview and some of the terminology of the language. Because the behavior of macros is different from that of code that is written for base SAS, sections are also included on macro execution and how the SAS System sees and uses macros.

1.1 Macro Facility Overview

The *SAS Macro Facility* is a tool within base SAS software that contains the essential elements that enable you to use macros. The macro facility contains a *macro processor* that translates macro code into statements that can be used by the SAS System, and the macro language. The *macro language* provides the means to communicate with the macro processor.

The macro language consists of its own set of commands, options, syntax, and compiler. While macro statements have similarities to the statements in the DATA step, you must understand the differences in behavior in order to effectively write and use macros.

The macro language provides tools that

- pass information between SAS steps
- dynamically create code at execution time
- conditionally execute DATA or PROC steps
- create generalizable and flexible code.

The tools made available through the macro facility include macro (or symbolic) variables, macro statements, and macro functions. These tools are included as part of the SAS code, or program, where they are detected when the code is sent to the SAS Supervisor for execution.

1.2 Terminology

The statement and syntax structure that is used by the macro facility is known as the *macro language* and like any language it has its own terminology. The SAS user familiar with the programming language used in base SAS, however, will discover quickly that

much of the syntax and content of the macro language is familiar.

The following terms will be used throughout this book.

text
a collection of characters and symbols that can contain variable names, data set names, SAS statement fragments, complete SAS statements, or even complete DATA and PROC steps.

macro variable
the names of macro variables are almost always preceded by an ampersand (&) in SAS code. Macro variables often are used to store text.

macro
stored text that contains SAS statements and macro language statements.

macro program statement
controls what actions take place during the macro execution. They are always preceded by a percent sign (%) and are often syntactically similar to statements used in the DATA step.

macro expression
one or more macro variable names, text, and/or macro functions combined together by one or more operators and/or parentheses. Macro expressions are very analogous to the expressions used in standard SAS programming.

macro function
predefined routines for processing text in macros and macro variables. Many macro functions are similar to functions used in the DATA step.

operators
symbols that are used for comparisons, logical operation, or arithmetic calculations. The operators are the same ones used in the DATA step.

automatic macro variable
special-purpose macro variables. These are automatically defined and provided by the SAS System. These variable names should be considered as reserved.

open code
SAS program statements that exist outside of any macro definition.

resolving macro references
during the resolution process, elements of the macro language (or references) are replaced with text.

You can find additional terminology in the glossary.

1.3 Macro Execution Phases

When you run a SAS program, it is executed in a series of DATA and PROC steps, one step at a time. GLOBAL statements (for example, TITLE, FOOTNOTE, %LET), which can exist outside of these steps, are executed immediately when they are encountered. For each step, the SAS System first checks to see if macro references exist. *Macro references* may be macro variables, macro statements, macro definitions, or macro calls. If the program does not contain any macro references, then processing continues with the DATA or PROC step processor. If the program

does contain macro references, then the macro processor intercepts and resolves them prior to execution. The resolved macro references then become part of the SAS code that is passed to the DATA or PROC step processor.

When code is passed to the SAS supervisor, the following takes place for each step:

- Global statements are executed.
- Macro definitions are compiled and stored until they are called.
- A check is made to see if there are any macro statements, macro variables, or macro calls. If there are, then
  - macro variables are resolved
  - called macros are executed (resolved)
  - macro statements are executed.
- The DATA or PROC step that contains resolved macro references (if there were any) is compiled and executed.

SEE ALSO

SAS® Macro Language Reference, First Edition contains a detailed discussion of how SAS processes statements with macro activity on pp.14–19 and 33–41.

1.4 Referencing Environments

Unlike the values of data set variables, the values of macro variables are stored in memory in a *symbol table*. Each macro variable's definition in the symbol table is also associated with a *referencing environment* or *scope*, which is determined by where and how the macro variable is defined. There are two environments for macro variables: global and local.

A *global* macro variable has a single value available to all macros within the program. Macro variables that are defined outside of any macro will be global.

Local macro variables have values that are available only within the macro in which they are defined.

Because each macro creates its own local referencing environment, macro variable values that are defined in one macro may be undefined within another. Indeed, macro variable names need not be unique even among nested macros. This means that the specific value associated with a given macro variable may depend on how the macro variable is used in the program.

In the following schematic, the macro variable DSN is defined globally and is, therefore, also known inside of the shaded macro. The macro variable COLOR, however, is only defined inside of the shaded macro and is not known outside of the macro.

```
Outside of all macros
Global values

DSN ---> clinics

Inside a macro
Local values

DSN ----> clinics
COLOR ---> blue

DSN ---> clinics
COLOR ---> undefined
```

You can control the referencing environment for a macro variable through the use of the %GLOBAL and %LOCAL statements, which are described in Section 5.4.2.

SEE ALSO

Extensive examples can be found in *SAS® Guide to Macro Processing, Version 6, Second Edition* (pp. 37–54) and the newer *SAS® Macro Language: Reference, First Edition* (pp. 50–66).

SUGI presentations that specifically cover referencing environments include Bercov (1993) and Hubbell (1990).

An example of a macro variable that takes on more than one value at the same time is given in Carpenter (1996, p. 1637).

1.5 Chapter Summary

You can think of the macro facility as a part of the SAS System that passively waits to be evoked. If your SAS code contains no macros and no references to macro variables or macro statements, the macro facility is not used. When the code does contain macro language references, the macro facility wakes up, intercepts the job stream, interprets or executes the macro references, and then releases its control.

The macro facility is made up of two primary components. The *macro processor* provides the ability to compile and execute the *macro language* statements that you use to write macros.