

Grab Your Tokens Here Comes the Macro Train

Mark MacMullen
MBNA Hallmark Information Services

Abstract

Do you know the four classes of tokens recognized by the SAS® word scanner? Do you understand how SAS handles tokenization of program text with the *Word Scanner*? Do you know what tokens trigger the *Macro Processor* to begin running?

This paper will address the process of compilation and execution of SAS programs from the *Input Stack* to the *Compiler* and the role of the *Macro Processor* and the *Symbol Table* plays for macro processing in conjunction with tokenization.

SAS users beginning to exploit the power of macros need to understand the tokenization that takes place during SAS execution. Comprehension of how the *SAS Word Scanner* divides program text into essential units called tokens helps to simplify writing and deciphering macro code. Whether you're starting out writing macrocode or supporting an existing application, this presentation will be of assistance to you.

Introduction

The fundamental aspects of the *Macro Facility* are often overlooked or glossed over without proper training. This paper will attempt to explain in detail those points that are crucial to a clearer understanding of the *Macro Facility*. Particular attention will be focused on the tokenization of the SAS language and its critical role in the solving of macrovariables, functions and statements.

Classes of Tokens

Program text that comes from the *Input Stack* into the *Word Scanner* is broken into basic units called tokens. The four classes of tokens are:

1. Literals - A string in single or double quotes. 'Alphanumeric' or "Alphanumeric"
2. Names - A string of characters beginning with a letter or underscore and continuing with letters, underscores or numbers.
file infile dollar10.2 format _n_
3. Numbers - A string of digits (integers). Date, time, datetime constants or hexadecimal constants.
99 100 '01aug1999'd .11 6.8E-4
4. Special Characters - Any character or grouping of characters that have a reserved meaning to the SAS compiler. **/ + - ** ; \$ () . & %*

A token ends when another token begins or a blank after a token.

The maximum length of a token in Version 6 is 200 characters and for Version 7 it is 32K. Once the programming text is changed to a token by the word scanner, it either goes to the compiler for execution or the macro processor. The two special characters or "triggers" needed for the macro processor to begin its work are the "%" or "&" when they are followed by a letter or underscore. Keep in mind that the macro processor is only one part of the macro facility, which is bundled in the Base SAS software and readily available to anyone with the courage to use it. The macro facility consists of the macro language, macro processor, symbol table and special macro handling routines.

Macro Simplicity

There is nothing really complex about the macro facility. The macro facility has only one primary fundamental truth from which you begin. *The macro facility is nothing more than a tool for text substitution.* This simple truth opens up so many countless possibilities. The only limitations to the macro facility are your imagination. From this humble beginning of text substitution you can do any of the following:

- Pass information between Data steps
- Pass information between macros
- Dynamically create SAS code at the time of execution
- Execute Data and Proc steps conditionally
- Create general and flexible SAS code
- Create and resolve macro variables anywhere in the SAS code
- Create data driven applications
- Hands free repetitive processing
- ...and much more

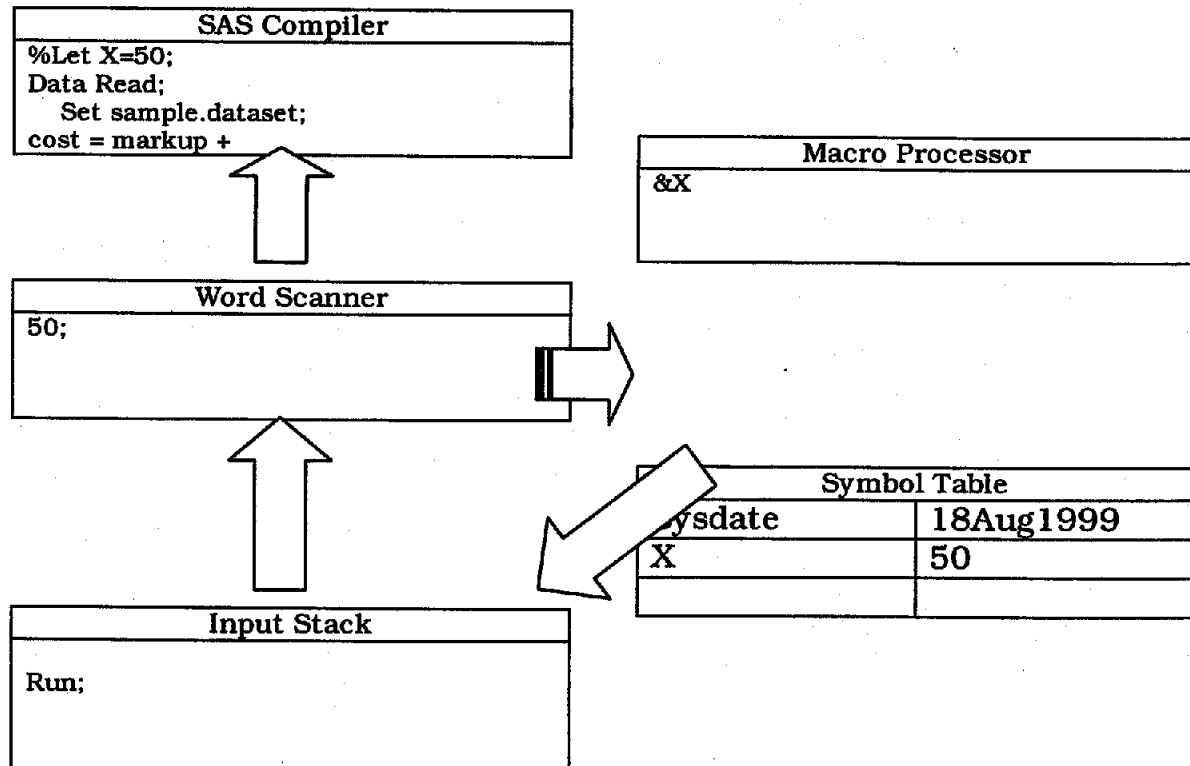
The programming tools of the macro facility consist of macrovariables, macrofunctions, macro keywords macro operators and macro statements and macro calls. Every macro statement begins with the "%" and a keyword and ends in a semicolon. Macro variables begin with the "&". When the SAS word scanner, (also known as the SAS Supervisor), sees either of these special characters the variable or statement is passed over to the macro processor. The macro processor then attempts to find the macrovariable in the symbol table. If the macrovariable cannot be resolved (cannot be found in the symbol table) then the macro processor hands the token back to the word scanner, which passes the *unresolved* macro to the compiler and the compiler writes an error message to the SAS log. Otherwise, the *resolved* macro

is passed back to the input stack, which in turn is passed onto the SAS word scanner for tokenization and on to the compiler for syntax checking and ultimately execution. Remember, any created macro is nothing more than text that has been substituted with a macro variable and the macro statements are nothing more than funny looking programming statements that follow the usual rules of programming logic. Admittedly there are more details that need to be learned, but macro processing never leaves these fundamental truths.

Tokenization and You

All SAS programs are compiled and executed in the same way. Execution of programming statements is done in the exact order they are encountered in the program. The word scanner receives the program text from the input stack and tokenizes each SAS language statement. The compiler then receives the tokens and does syntax checking and at a SAS step boundary executes the program. However, if the word scanner receives a macro "trigger" i.e. % or & followed by a letter or underscore, it sends the macro variable, statement, or call to the macro processor and the macro processor takes over control temporarily until a semicolon is detected. The macro variable, statement, or call is turned into SAS language and the results are stored in a memory location called the *symbol table* where other local and/or global macros may also be stored. Then the macro processor re-stacks the *resolved* macrovariable that has become SAS language text back on top of the input stack. The word scanner resumes its task of tokenizing the SAS language statements and passes them on to the compiler for syntax checking and execution. See the pictorial

demonstration below for further explanation.



Conclusion

This paper has dealt with one of the very basic issues of understanding how the SAS language processes program text including the addition of macro resolution. Tokenization is a principle part of the compilation process and when understood makes the planning, writing, and debugging of SAS coding incredibly easier. Macros can make your job as a programmer much easier and allow you to do things with SAS code that otherwise would be impossible. I would encourage anyone who has not taken the Instructor Based Training from SAS Institute on SAS Macro Language to do so. These fundamentals and a plethora of additional information will revolutionize your SAS coding experience. However, if you cannot get to this class the knowledge is still available to you through the Books by Users, which address this subject at the most basic of levels for anyone.

Trademark Information:

SAS is a trademark of SAS Institute Inc. in the United States and other countries. ® indicates USA registration.

References:

SAS Institute Inc., *SAS Macro Language, Course Notes*, Book Code 58213, course code MACR, Cary, NC: SAS Institute Inc. 1998. 379pp.

Aster, Rick, and Seidman, Rhena, *Professional SAS Programming Secrets*, The McGraw-Hill Companies, Inc., 1997. 588pp.

Burlew, Michele M., *SAS Macro Programming Made Easy*, Cary, NC: SAS Institute Inc., 1998. 280pp.

Carpenter, Art, *Carpenter's Complete Guide to the SAS Macro Language*, Cary, NC: SAS Institute Inc., 1998. 242pp.

Author:

The author can be contacted at:
MBNA Hallmark Information Services
P.O. Box 809051
Dallas, Texas 75380-9051
972-701-6308