

SAS® PROGRAMMING STANDARDS OR NOBODY'S GONNA TELL ME WHAT TO DO!

John Olson, McDonnell Douglas Helicopter Systems and
Michael L. Sperling, City of Phoenix, AZ

ABSTRACT

The subject of standards has been a debatable issue since the days when Adam was told not to eat the apple (Not the first computer). What kind of standards are acceptable and to whom? To whom should they apply...end-users, programmers, production jobs only? What is the difference between a standard and a guideline? A suggested standard will be distributed for discussion. This will be an audience participation presentation.

The views below are to generate discussion and are not necessarily the opinions of the presenters. The attached exhibit is a sample standard for SAS programs and is the basis of this presentation.

Standards:

- 1) inhibit creativity;
- 2) promote consistency;
- 3) establish communications among programmers;
- 4) are a device by which programmers can communicate effectively;
- 5) are restrictive;
- 6) make maintenance easier;
- 7) cause development to take too much time;
- 8) are developed by people who are not practitioners;
- 9) are developed by the SAS experts;
- 10) WILL be followed;
- 11) should be followed, only if reasonable;
- 12) only apply to people who write bad code (other people);
- 13) restrict bad practices;
- 14) restrict all practices;
- 15) create documentation which is insufficient;
- 16) create documentation;
- 17) apply to all SAS programmers;
- 18) apply only to end user programmers;
- 19) support creativity;
- 20) support the bureaucracy;
- 21) support the entire enterprise;
- 22) are too confusing.

ACKNOWLEDGMENTS

Valley of the Sun SAS Users Group

SAS is a registered trademark of SAS Institute Inc. In the USA and other countries.

® indicates USA registration.

John Olson, McDonnell Douglas Helicopter Systems, Mesa, AZ.

Phone: (602) 891-3963

Fax: (602) 891-7958

E-mail: JOLSON AZ@AOL.COM

Michael Sperling, City of Phoenix, Arizona

Phone: (602) 256-4190

Fax: (602) 534-1488

E-mail: msperlin@ci.phoenix.az.us

**EXHIBIT
SUGGESTED STANDARD**

1.0 PURPOSE

This standard will guide information systems personnel in programming more efficient and easily maintainable self-documenting code.

2.0 SCOPE

- 2.1 This standard applies to all new in-house-developed programs. Programs which were substantially or entirely coded prior to the publication of this standard need not conform except to the extent that new code within the existing code must be written according to this standard.
- 2.2 This standard should be considered when reviewing program performance and computer resource utilization.
- 2.3 This standard is not binding on user programmers developing SAS code outside the Information Systems Group, but it may be used as a reference for SAS development.

3.0 ENFORCEMENT

- 3.1 The manager, Applications Development, will be responsible for monitoring and enforcing the use of this standard.
- 3.2 If noncompliance is detected, the Manager will reject the turnover of code.

4.0 GENERAL

Conventions for file naming, code formatting and appropriate use of particular SAS statements or variables are essential to permit the interchange of programs which can be easily understood. These conventions are linked to one of the following major areas.

4.1 Maintenance

- A. If a DATA step uses the same constant frequently, substitute a variable initialized in a RETAIN statement. This will allow the constant value to be changed with only one modification instead of many, thereby enhancing efficiency.
- B. When multiple datasets are created in a single data step, use in-line comments to describe each dataset created as well as the functions of the data step.
- C. When statements are used to generate diagnostic printout, they should be clearly identified with the in-line comment /*TEST*/. When the diagnostic printout is no longer needed, convert these statements to SAS comments.
- D. When using a conditional DO loop (indexed DO, DO WHILE or DO UNTIL) after a THEN or ELSE statement, nest it within an unconditional DO loop.

EXHIBIT SUGGESTED STANDARD

4.2 Documentation

A. Housekeeping

1. Program Header

The following format will be used for all SAS program headers. The header will be used to provide an overview of the objectives of the program to include the maintenance of the change log.

=====

Function: A brief overview of the functions and objectives

Files Use:

FD Name	Opened	Type	Description
---------	--------	------	-------------

Calling Programs

Special Requirements

Date	PGMR Name	Cntl #	Change Note
------	-----------	--------	-------------

1/1/95	NNNNN	XXXXXX	Why Change?
--------	-------	--------	-------------

1/1/99	NNNNN	XXXXXX	Why Change?
--------	-------	--------	-------------

=====

2. EOJ Display

- a. SAS default will display counts at the end of each data or procedural step indicating the number of records read and written from each file. Any additional displays which were added for debugging should be removed prior to production turnover.

3. Programmed ABENDS

- a. System option ERRORABEND is the default system option and must be included in to avoid the deletion of input files and cataloging of erroneous output files.
- b. Use the standard SAS ABORT ABEND N to force a user specified ABEND code to abnormally terminate the job.
- c. Use the standard SAS ABORT RETURN N to cause an immediate normal termination of the SAS job or step.

EXHIBIT SUGGESTED STANDARD

- B. Naming Conventions** - It is extremely important to have meaningful, descriptive names for all variables within the program.
1. **Program Names** - Use same name as output dataset (if possible) so readers can tell which program created which dataset.
 2. **Format Names** - Use same as variable name being formatted (if only one) or variable name plus FMT.
 3. **IN = varname** - Naming of IN variables should be 'IN' plus dataset abbreviation.
 4. **Use labels** - Variable and dataset labels extend your ability to convey entity meaning.
 5. **Arrays** - Name all ARRAYS with a single UNDERSCORE '_' at the end so array names can be distinguished immediately from variable names. The first part of the ARRAY name should relate to the variables included in the array.
- C. Readability**
1. Use only one source code statement per line.
 2. Use a standard and consistent indentation scheme. It is recommended that all PROC and DATA statements begin in column one. Any other statements should be indented by 2 or more spaces. The SAS statement following an ELSE should be on the next line and indented 2 spaces. All statements between DO and END statements should also be indented 2 spaces with respect to the DO statement.
 3. Line up all DO statements with their corresponding END statements and put in-line comments on the same line. These comments will assist in identifying the proper loop when loops are nested.
 4. To delimit/clarify DATA or PROC steps, use a RUN statement after the final executable statement and at least one blank line between steps. The RUN statement will force notes generated during execution to be printed on the log within the step that generated them (not mandatory).
 5. The beginning of each block of code should be clearly identified with comments, especially to clarify unusual or complex logic.
 6. All non-executable statements in a DATA step should follow the DATA statement and should precede all executable statements. The first non-executable statement should be the length statement, if it is used.
 7. When using the END = option on a SET, MERGE or UPDATE statement, set END = LASTREC to emphasize the variable has a value of 1 only for the final record.
 8. Use option S=72. This places all code within the normal edit screen. All problems caused by mixing code with sequence numbers and no sequence numbers are prevented.

EXHIBIT SUGGESTED STANDARD

4.3 Efficiency

- A. Avoid mixed data type calculations. SAS will convert character variables to numeric for calculation, but it consumes a surprising amount of CPU time to do it. Use the INPUT function to perform conversions prior to the arithmetic operations.
- B. Use SUM statements rather than RETAIN statements with assignment statements to accumulate totals in the data step (i.e., TOTAL + COST).
- C. In an ARRAY statement, the index variable should be defined explicitly. When processing the array within a DO loop, the indexed DO loop should be used instead of the DO OVER 'array name' loop.
- D. Use standard SAS procedures or functions in lieu of developing your own code.
- E. Datasets should be deleted when no longer needed if the deletions would result in a reduction in work space significant enough to offset the Proc Delete cost.
- F. See SAS manuals for specific information on processing large datasets.

4.4 Error Prevention and Debugging

- A. Declare character variables using the length statement. This avoids the trap of a variable defined numeric by default.
- B. For table lookups or recoding, use a PROC FORMAT with a PUT function if there are more than 2-10 possible codes. It is much easier to change the values in a format which will serve an entire program than it is to modify code resident in each DATA step of a large program.
- C. For nested IF-THEN-ELSE-IF, arrange the conditions in order from most likely to least likely. In most cases, this will result in fewer statements being executed.
- D. It is recommended that any statement which uses values of the FIRST., LAST., IN= or END= variables be executed before any DELETE, RETURN, or subsetting IF statements. (Caution: failure to comply may produce inaccurate results.)
- E. Always use DATA = 'dataset name' in PROC statements and name datasets explicitly in SET statements to ensure that the dataset used by a PROC is the one intended. AVOID default dataset names.

**EXHIBIT
SUGGESTED STANDARD**

4.5 Logic

- A. Benign errors not allowed include missing values used in calculations, variables not initialized, unknown macro variables, mathematical operations not performed, data too wide for format, etc. The program can have an internal comment or produce an explanatory message to reassure the user that 'error' does not require correction. Reconcile the log to output; log messages may indicate other unknown problems.
- B. Variables without specific purpose (not used within the program) should not be created.
- C. Exception handling - Program should detect violations of the assumptions about its input.
- D. Structured code - Avoid use of GO-TO statements and other non-structured constructs.
- E. Merging requires a BY statement. Avoid using the same variables on the different datasets being merged (except for BY variables); they will overlay each other.
- F. Speed of execution - Balance processing efficiency with logical simplicity in situations where they may conflict.
- G. Minimize interaction between programs and steps. Avoid having one program do the work for another. Make programs self-sufficient if possible.
- H. Standardize the sorted order of datasets if helpful.
- I. Cause/effect distance - Minimize the distance between (1) causes and effects, (2) problems and solutions, and (3) comments and code being explained.
- J. Don't over code - Exploit SAS's higher level language capability. Use the output dataset capabilities of procedures.
- K. Use PROC SQL in lieu of merge statements, when a many-to-many condition exists, to eliminate errors in the output data set.