Don't Despair, You Do Have An Option!

Wei (Lisa) Lin, Jiannan (Jane) Kang
Merck & Co., Inc., Upper Gwynecdd, PA

ABSTRACT

A large number of SAS® programmers may never change SAS system options. Why? Because SAS provides a full load of default system options based on assumptions of how programmers want the system to work. Default system options save programmers from having to specify every little detail when working in SAS, but programmers are not always satisfied with the assumptions SAS formulates. However, SAS provides a way for programmers to override the default options. Some options can only be specified when SAS initializes, but most system options can be changed at any time during the SAS session with a simple OPTIONS statement. This paper will discuss some of the common SAS system options that will make SAS work easier.

INTRODUCTION

SAS system options control how the SAS System formats output, handles files, processes data sets, interacts with the operating environment, and performs other tasks that are not specific to a single SAS program or data set. Changes to the settings of SAS system options are completed through following methods:

- In the SAS command;
- In a configuration or autoexec file;
- In the SAS OPTIONS statement;
- By using the OPTLOAD and OPTSAVE procedures;
- Through the SAS System Options window;
- In other ways, depending on the operating environment.

If the same system option appears in more than one place, the order of precedence from highest to lowest is as follows:

1. OPTIONS statement and SAS System Options window
2. autoexec file (that contains an OPTIONS statement)
3. command-line specification
4. configuration file specification
5. SAS system default settings.

When specifying a SAS system option setting, the setting applies to the next step and to all subsequent steps for the duration of the SAS session or until a reset is applied. A list of the available system options can be viewed by using the OPTIONS procedure or by going to VOPTION dictionary tables located in the SASHELP library. To use the OPTIONS procedure, submit the following SAS statements and view the results in the SAS log:

```
PROC OPTIONS;
RUN;
```

In this paper, SAS OPTIONS statement is discussed as a way of changing system options. The syntax for specifying system options in an OPTIONS statement is:

```
OPTIONS option(s);
```

Here 'option' specifies one or more SAS system options under consideration for change. In these statements, keyword options can appear simply as the option name to turn the option on, or the option name prefixed by NO to turn the option off. For keyword-value options, the option name is followed by an equal sign (=) and an appropriate value for the option.

A few options can take either a value or the NO prefix; in that case, the NO prefix is equivalent to a blank value or no value assigned to the option.

There are over 300 options available with SAS version 9. They are grouped into 23 groups based on the following functionality:

**COMMUNICATIONS**
- DATAQUALITY
- ERRORHANDLING
- EMAIL
- ENVDISPLAY

**ENVFILLES**
- HELP
- INPUTCONTROL

**GRAPHICS**
- LISTCONTROL
- LOG_LISTCONTROL

**LANGUAGECONTROL**
- MEMORY
- META

**MACRO**
- SASFILES
- SORT

**PERFORMANCE**
- EXECMODES
- INSTALL
- LOGCONTROL
- ODSPRINT
- EXTFILES
To display the settings of system options with a specific functionality, use the `GROUP=` option as shown below:

```sas
PROC OPTIONS GROUP=ERRORHANDLING;
RUN;
```

### SOME USEFUL SAS SYSTEM OPTIONS BY GROUP

#### ENVDISPLAY (Environment control: Display)

- **OPTIONS NOCHARCODE | CHARCODE**
  
  This option specifies whether to use character combinations as substitutes for special characters not on the keyboard. You can find those character combinations in SAS Help document.

  **Example:**
  
  ```sas
  OPTIONS CHARCODE;
  title '?( test title ?= test title  ?/ test title ?>';
  ```

  This statement produces the output:

  `{test title ^ test title  | test title }

- **ERRORHANDLING (Environment control: Error Handling)**

  - **OPTIONS DSNFERR**
    
    **DSNFERR** controls how SAS responds when a SAS data set is not found.

    **Example:**
    
    ```sas
    OPTIONS DSNFERR;
    data b; set a; run;
    Log:
    ERROR: File WORK.A.DATA does not exist.
    NOTE: The SAS System stopped processing this step because of errors.
    WARNING: The data set WORK.B may be incomplete. When this step was stopped there were 0 observation and 0 variable.
    ```

    ```sas
    OPTIONS NODSNFERR;
    data b; set a; run;
    Log:
    NOTE: The data set WORK.B has 0 observation and 0 variable.
    ```

  - **OPTIONS NODMSSYNCHK | DMSSYNCHK**
    
    **NODMSSYNCHK** does not enable syntax check, in windowing mode, for a submitted statement block. If **NODMSSYNCHK** is in effect, SAS processes the remaining steps even if an error occurs in the previous step. **DMSSYNCHK** is to validate syntax in an interactive session by using the SAS windowing environment.

    **Example:**
    
    ```sas
    OPTIONS DMSSYNCHK;
    data a; a=1; run;
    OPTIONS NODMSSYNCHK;
    data test;
    A=1; if a=.  b=1;
    run;
    Proc print data=a; run;
    Log:
    184  a=1; if a=.  b=1;
    22
    ERROR 22-322: Syntax error, .......
    NOTE: The SAS System stopped processing this step because of errors.
    WARNING: The data set WORK.TEST may be incomplete. When this step was stopped there were 0 observation and 2 variables.
    ```
WARNING: Data set WORK.TEST was not replaced because this step was stopped.

186 proc print data=a; run;

NOTE: PROCEDURE PRINT used (Total process time):
Output: There is no output display for PROC PRINT procedure in OUTPUT window.

OPTIONS NODMSSYNCHK;
Data test;
   A=1; if a=. b=1;
Run;
Proc print data=a; run;
Log: With system options NODMSSYNCHK enabled, in addition to the notes, error, and the warning messages above, SAS reports a note from PROC PRINT procedure. SAS outputs the data set a in OUTPUT window.

192 proc print data=a; run;

NOTE: There were 1 observation read from the data set WORK.A.

NOTE: PROCEDURE PRINT used (Total process time):

- OPTIONS ERRORCHECK = NORMAL | STRICT
  This system option is used with DMSSYNCHK system option. It is used to specify the syntax check mode for the LIBNAME statement, the FILENAME statement, the %INCLUDE statement, and the LOCK statement in SAS/SHARE.
  NORMAL does not place the SAS job into syntax-check mode when an error occurs in the above example.
  STRICT places the SAS job into syntax-check code when an error occurs in the above example.
  Example:
  OPTIONS DMSSYNCHK ERRORCHECK = NORMAL;
  libname - newlib 'c:\temp';

Log:
ERROR: - is not a valid SAS name.
ERROR: Error in the LIBNAME statement.

data b1 ; set a; run;
Log:
NOTE: There were 1 observation read from the data set WORK.A.

NOTE: The data set WORK.B1 has 1 observation and 1 variable.

OPTIONS DMSSYNCHK ERRORCHECK = STRICT;
libname - newlib 'c:\temp';
Log:
ERROR: - is not a valid SAS name.
ERROR: Error in the LIBNAME statement.

data b2 ; set a; run;
NOTE: The data set WORK.B2 has 0 observation and 1 variable.

- OPTIONS ERROR =
  ERROR = controls the maximum number of observations for which complete error messages are printed.
  Example:
  OPTIONS ERRORS=2;
data test1;
   infile 'u:\options\test.dat';
   input name 1-10 ;
run;
Log:
NOTE: The infile 'u:\options\test.dat' is:
         File Name=u:\options\test.dat, RECFM=V,LRECL=256
NOTE: Invalid data for name in line 1 1-10.
RULE:     ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----
---+----8----+-
1  jane       5  f 17
name=. _ERROR_=1 _N_=1
NOTE: Invalid data for name in line 2 1-10.
ERROR: Limit set by ERRORS= option reached. Further errors of this type will not
be printed.
2  mary      12  f 17
name=. _ERROR_=1 _N_=2
NOTE: 3 records were read from the infile 'u:\options\test.dat'.
The minimum record length was 17.
The maximum record length was 17.
NOTE: The data set WORK.TEST1 has 3 observations and 1 variables.

• OPTIONS INVALIDDATA = 'character'
  INVALIDDATA = system option specifies the value that SAS is to assign to a variable when invalid numeric
  data are read with an INPUT statement or the INPUT function; it can be a letter (A to Z, a to z), a period(.), or
  an underscore (_). The default value is a period.
  Example:
  OPTIONS INVALIDDATA='.';
  data test2;
     infile 'u:\options\test1.dat';
     input name $ 1-10 age 11-15 gender $ 16-17;
  proc print; run;
  Output:
    Obs name age gender
    1 jane 5 f
    2 mary 12 f
    3 tim 7 m
    4 john 10 m
    5 jenny _ f
    6 jim _ m

❖ LISTCONTROL (Log and procedure output control: Procedure output)

• OPTIONS FORMDLIM = 'delimiting-character';
  FORMDLIM specifies in quotation marks a character written to delimit pages. Normally, the delimit character is
  null. When the delimit character is null, a new physical page starts whenever a new page occurs. However, it is
  possible to conserve paper by allowing multiple pages of output to appear on the same page. For example, this
  statement writes a line of dashes (- -) where normally a page break would occur: options FORMDLIM ='-';
  When a new page is to begin, SAS skips a single line, writes a line consisting of the dashes that are repeated
  across the page, and skips another single line. There is no skip to the top of a new physical page. Resetting
  FORMDLIM to null causes physical pages to be written normally again.
  Example:
  options formdlim='%';
  title "dataset test4, delimit character %";
  proc print; run; proc print; run;
  Output:
• **OPTIONS BYSORTED | NOBYSORTED**

    **BYSORTED** specifies that observations in a data set, or data sets are sorted in alphabetic or numeric order. **NOBYSORTED** specifies that observations with the same BY value are grouped together but are not necessarily sorted in alphabetic or numeric order.

    When the **NOBYSORTED** option is specified, stating NOTSORTED on every BY statement to access the data set(s) is not required.

    **NOBYSORTED** is useful when data falls into other logical groupings such as chronological order or linguistic order thereby allowing BY processing to continue without failure when a data set is not actually sorted in alphabetic or numeric order.

**Example:**

```sas
data test;
a='a'; b=1; c='f'; output;
a='b'; b=2; c='f'; output;
a='b'; b=3; c='f'; output;
a='a'; b=4; c='m'; output;
a='a'; b=5; c='m'; output;
a='b'; b=6; c='g'; output;
a='a'; b=7; c='g'; output;
run;
```

This data set "test" is not sorted before executing the following proc print step with BY statement, but it is grouped by variable c and variable a alphabetically. In group (c=g), variable a is not ordered in alphabetic order.

```sas
title "System Option BYSORTED in effect without BY Statement Option ";
OPTIONS BYSORTED;
proc print data=test;
   by c a;
run;
```

**Log:** SAS reports error message running the proc print step, and only 6 observations are read because variable a is not sorted by alphabetic order in the last group (c=g).

**ERROR:** Data set WORK.TEST is not sorted in ascending sequence. The current by-group has c = m and the next by-group has c = g.

**NOTE:** The SAS System stopped processing this step because of errors

**NOTE:** There were 6 observations read from the data set WORK.TEST.

**NOTE:** PROCEDURE PRINT used (Total process time):

**Output:**

Only the first group of value 'c=f' is printed in the output window.
System Option BYSORTED in effect without BY Statement Option

10:31 Monday, June 15, 2009

title "System Option BYSORTED in effect with BY Statement Option ";
proc print data=test;
  by c a NOTSORTED ;
run;
Log:
NOTE: There were 7 observations read from the data set WORK.TEST.
NOTE: PROCEDURE PRINT used (Total process time):
Output: With system option BYSORTED and BY statement option NOTSORTED in effect, data are printed out by value group and by variable c and variable a.

System Option NOBYSORTED in effect without BY Statement Option

10:31 Monday, June 15, 2009

title "System Option NOBYSORTED in effect with BY Statement Option ";
OPTIONS NOBYSORTED;
proc print data=test;
  by c a;
run;
Log:
NOTE: There were 7 observations read from the data set WORK.TEST.
NOTE: PROCEDURE PRINT used (Total process time):
SAS outputs the same exact data as the BY statement option above. With system option NOSORTED in effect, the BY statement options are no longer needed.

❖ LOG_LISTCONTROL (Log and procedure output control: SAS log and procedure output)

- OPTIONS DATE | NODATE
  DATE prints the date and time the SAS job began at the top of each page of the SAS log and output, this is the default.

Page 6 of 13
• **OPTIONS LINESIZE= n| MIN | MAX| hexX**
  This option specifies the line size (printer line width) for the SAS log and output used by the DATA step and procedures; n specifies the number of lines (default is 96); MIN sets the line size to minimum of 64; MAX sets the line size to maximum of 256.

• **OPTIONS MISSING='character'**
  This option specifies to print character value to represent missing value. Single or double quotation marks are optional. The period is the default. If you use other character, SAS requires an option value that contains a maximum of 1 character.
  **Example:**
  OPTIONS MISSING=' ';  
  Data test;  
  Format a $1. b 8.;  
  A='1'; output;  
  Run;  
  Proc print; run;  
  Output:  
  Obs a b  
  1 1  

• **OPTIONS NUMBER | NONUMBER**
  NUMBER specifies that SAS prints the page number on the first title line of each page of SAS output.

• **OPTIONS PAGESIZE= n| nK | hexX | MIN | MAX**
  n specifies the number of lines that compose a page. The default is 54 lines. MIN sets the number of lines to the minimum setting of 15. MAX - sets the number of lines to the maximum setting which is 32,767.

❖ **LOGCONTROL (Log and procedure output control: SAS log)**

• **OPTIONS MSGLEVEL= N | I**
  N prints notes, warnings, and error messages only; this is the default.
  I prints additional notes pertaining to index usage, merge processing, sort utilities, and CEDA usage, along with standard notes, warnings, and error messages. It is especially useful to know if the merge is performed as requested. If MSGLEVEL=I, SAS writes a warning to the SAS log when a MERGE statement would cause variables to be overwritten.
  **Example:**
  data test1;  
  name='Jane'; age=10; class='3rd grade'; output;  
  run;  

  data test2;  
  name='Jane'; age=9; time='9:00am'; output;  
  run;  

  OPTIONS MSGLEVEL= N;  
  data test;  
  merge test1 test2; by name;  
  run;  
  Log:  
  NOTE: There were 1 observation read from the data set WORK.TEST1.
NOTE: There were 1 observation read from the data set WORK.TEST2.
NOTE: The data set WORK.TEST has 1 observation and 4 variables.

OPTIONS MSGLEVEL= I;
Log:
INFO: The variable age on data set WORK.TEST1 will be overwritten by data set WORK.TEST2.
NOTE: There were 1 observation read from the data set WORK.TEST1.
NOTE: There were 1 observation read from the data set WORK.TEST2.
NOTE: The data set WORK.TEST has 1 observation and 4 variables.
This log provides useful information if an additional variable is needed to be merged by since there might be several students with the same name or data issues regarding age (since one shows age 10, and the other data set shows age 9).

❖ MACRO (MACRO: SAS macro)

- OPTIONS MAUTOLOCDISPLAY | NOMAUTOLOCDISPLAY
MAUTOLOCDISPLAY enables the MACRO to display the autocall macro source location in the log when the autocall macro is invoked.

- OPTIONS MAUTOSOURCE | NOMAUTOSOURCE
MAUTOSOURCE causes the macro processor to search the autocall libraries unless the macro name is found in the WORK library.

- OPTIONS SASAUTOS= library-specification | (library-specification-1,...,library-specification-n)
With this option it is possible to identify one or more locations of library members that contain a SAS macro definition. A location can be a SAS fileref or a host-specific location name enclosed in quotation marks. When specifying two or more autocall libraries, enclose the specifications in parentheses and separate them with either a comma or a blank space. When SAS searches for an autocall macro definition, it opens and searches each location in the same order that it is specified in the SASAUTOS option.

Example:
The following code is to setup SAS autocall in the sequence of: 1. sasautos; 2. protocol level macros; 3. project level macros; 4. global level macros with mautolocdisplay option.

OPTIONS MAUTOSOURCE MAUTOLOCDISPLAY SASAUTOS =(
sasautos
   /* --- Protocol level macros --- */
fptmla  fptmlb  fptmlc
   /* --- Project level macros --- */
Fpjmla  fpjmlb  fpjmlc
   /* --- Global level macros --- */
fglmla  fglmlb  fglmlc );

Below are some examples of logs after a macro is run; MAUTOLOCDISPLAY enables MACRO to display the autocall macro source location.
316   %mkadrparm( input_dataset=RDADRPARM,
MAUTOLOCDISPLAY(MKADRPARM):  This macro was compiled from the autocall file
\wpmrldata53\SDE\macrolib\SDD\Prod\library\macros\ADaM\mkadrparm.sas
317   output_dataset=adrparm,
318   debug=N)

- OPTIONS MLOGIC | NOMLOGIC
MLOGIC causes the macro processor to trace its execution and to write the trace information to the SAS log. This option is a useful debugging tool, and it displays messages that identify the following:
  o  the beginning of macro execution
- values of macro parameters at invocation
- execution of each macro program statement
- whether each %IF condition is true or false
- the ending of macro execution.

- OPTIONS SYMBOLGEN | NOSYMBOLGEN
  SYMBOLGEN displays the results of resolving macro variable references. SYMBOLGEN displays the results in this form "SYMBOLGEN: Macro variable name resolves to value". SYMBOLGEN also indicates when a double ampersand (&&) resolves to a single ampersand (&).

- OPTIONS MPRINT | NOMPRINT
  MPRINT displays the SAS statements that are generated by macro execution.

Typically, the options MLOGIC, MPRINT and SYMBOLGEN are used together to debug macros. Please note using these options can produce a great deal of output.

Example:
If using OPTIONS MLOGIC MPRINT SYMBOLGEN for debugging purposes, following is the log with detail information about the macro compiling and execution:
SYMBOLGEN: Macro variable DEBUG resolves to N
MLOGIC(MKADRPARM): %IF condition "&debug"="Y" is FALSE
MPRINT(MKADRPARM): options compress=yes;
MLOGIC(MKADRPARM): %LET (variable name is VARS)
MLOGIC(MKADRPARM): %LET (variable name is _VARS)
MLOGIC(DSVAR): Beginning execution.
MLOGIC(DSVAR): This macro was compiled from the autocall file \wpmrldata53\SDE\statTEST\MK869\Emesis\PROT130Cycle1\macrolib\utility\dsvar.sas
SYMBOLGEN: Macro variable INPUT_DATASET resolves to WORK.RDADRPARM

- OPTIONS MLOGICNEST | NOMLOGICNEST
  MLOGICNEST displays the macro nesting information in the MLOGIC output in the SAS log. The setting of MLOGICNEST does not imply the setting of MLOGIC. It is required to turn on both MLOGIC and MLOGICNEST in order for output (with nesting information) to be written to the SAS log.

Example:
%macro test1;
  %put THIS IS TEST1;
%mend;
%macro test2;
  %put THIS IS TEST2;
  %test1;
%mend test2;
%macro test3;
  %put THIS IS TEST3;
  %test2;
%mend test3;

OPTIONS MLOGIC MLOGICNEST;
%test3

Log:
MLOGIC(TEST3): Beginning execution.
MLOGIC(TEST3): %PUT THIS IS TEST3
THIS IS TEST3
MLOGIC(TEST3.TEST2): Beginning execution.
MLOGIC(TEST3.TEST2): %PUT THIS IS TEST2
THIS IS TEST2
MLOGIC(TEST3.TEST2.TEST1): Beginning execution.
MLOGIC(TEST3.TEST2.TEST1): %PUT THIS IS TEST1
   THIS IS TEST1
MLOGIC(TEST3.TEST2.TEST1): Ending execution.
MLOGIC(TEST3.TEST2): Ending execution.
MLOGIC(TEST3): Ending execution.

❖ ODSPRINT (Log and procedure output control: ODS printing)

- **OPTIONS BOTTOMMARGIN= | TOPMARGIN= | LEFTMARGIN= | RIGHTMARGIN=**
  These options tell a printer the size of the margin at the top, bottom, left and right of the page. The default setting of BOTTOMMARGIN and TOPMARGIN is 0.2 IN, LEFTMARGIN and RIGHTMARGIN is 0.25 IN. Changing the value of these options may result in changes to the value of the PAGESIZE= and LINESIZE= system option.
  **Example:**
  OPTIONS BOTTOMMARGIN="0.05 IN" LEFTMARGIN="0.1 IN" ;

- **OPTIONS ORIENTATION=PORTRAIT | LANDSCAPE | REVERSEPORTRAIT | REVERSELANDSCAPE**
  PORTRAIT|LANDSCAPE specifies the paper orientation as portrait or landscape; "Portrait" is the default.
  REVERSEPORTRAIT| REVERSELANDSCAPE specifies the paper orientation as reverse portrait or landscape to enable printing on paper with prepunched holes.

❖ SASFILES (Files: SAS Files)

- **OPTIONS COMPRESS=NO | YES | CHAR | BINARY**
  NO specifies that the observations in SAS data set are uncompressed (fixed-length records); this is the default.
  YES | CHAR specifies that the observations in SAS data set are compressed (variable-length records) by reducing repeated consecutive characters (including blanks) to two-byte or three-byte representations.
  BINARY is highly effective for compressing medium to large (several hundred bytes or larger) blocks of binary data (numeric variables).
  Compressing a file reduces the number of bytes required to represent each observation and therefore reduces storage requirements for the file and saves time to process data. The tradeoff is an increased CPU requirement because the CPU has to uncompress each observation before processing them. Often OPTIONS COMPRESS=YES is used to process lab, vital or other large data sets to save processing time.
  **Example:**
  Following is an example of doing a simple sort for a lab data set with two different options:
  OPTIONS COMPRESS=NO;
  proc sort data=lptss.lb out=lb;
      by usubjid subjid;
  run;

  **Log:**
  NOTE: There were 120842 observations read from the data set LPTSS.LB.
  NOTE: The data set WORK.LB has 120842 observations and 74 variables.
  NOTE: PROCEDURE SORT used (Total process time):
       real time   4:54.41
       cpu time   11.50 seconds

  OPTIONS COMPRESS=YES;

  **Log:**
  NOTE: There were 120842 observations read from the data set LPTSS.LB.
  NOTE: The data set WORK.LB has 120842 observations and 74 variables.
  NOTE: Compressing data set WORK.LB decreased size by 97.96 percent.
  Compressed is 2469 pages; un-compressed would require 120842 pages.
  NOTE: PROCEDURE SORT used (Total process time):
       real time   2:17.88
       cpu time   9.32 seconds
As you can see from the log comparison above, the processing time with COMPRESS=YES has been reduced by more than 50%.

- **OPTIONS FIRSTOBS= MIN | MAX | nK | nM | nG | nT | hex;**
  MIN sets the number of the first observation to minimum value which is 1; this is the default.
  n (integer number) is used to process data from nth observation.

- **OPTIONS OBS= n | nK | nM | nG | nT | hexX | MIN | MAX;**
  n (integer number) indicates when to stop processing observation or records. MAX is about 9.2 quintillion; this is the default.
  While the FIRSTOBS= system option specifies a starting point for processing, the OBS= system option specifies an ending point. The two options are often used together to define a range of observations or records to be processed. To determine when to stop processing, SAS uses the formula of \((obs - firstobs) + 1 = results\). For example, if OBS=10 and FIRSTOBS=1 (which is the default for FIRSTOBS=), the result is 10 observation or records, that is \((10 - 1) + 1 = 10\).

**Examples:**
```sas
data old (drop=i);
  do i=1 to 100;
    A=i; output;
  end;
run;

OPTIONS FIRSTOBS=11 OBS=50;
data a;
  set old; /* 100 observation */
run;
data b; set a; run;
data c; set b; run;
Log:
NOTE: The data set WORK.OLD has 100 observation and 1 variable.
NOTE: The data set WORK.A has 40 observation and 1 variable.
NOTE: The data set WORK.B has 30 observation and 1 variable.
NOTE: The data set WORK.C has 20 observation and 1 variable.
```

- **OPTIONS MERGENOBY= NOWARN | WARN | ERROR**
  NOWARN specifies that no warning message is issued; this is the default.
  ERROR specifies that an error message is issued.
  Merging with no "BY statement" is dangerous unless it is a One-to-One merging (combines the first observation from all data sets that are named in the MERGE statement into the first observation in the new data set, the second observation from all data sets into the second observation in the new data set, and so on). Usually a Match-merging is performed which combines observations from two or more SAS data sets into a single observation in a new data set according to the values of a common variable - definitely the "BY statement" is required, and this option should be set to WARN or ERROR.

**Example:**
```sas
data test1;
  name='Jane'; age=10; class='3rd grade'; output;
  name='Amy'; age=12; class='5th grade'; output;
run;

data test2;
  name='Jane'; age=9; time='9:00am'; output;
  name='Jane'; age=10; time='8:45am'; output;
  name='Amy'; age=12; time='8:30am'; output;
run;
```
OPTIONS MERGENOBY= NOWARN;
data test;
    merge test1 test2;
run;

Log:
NOTE: There were 2 observations read from the data set WORK.TEST1.
NOTE: There were 3 observations read from the data set WORK.TEST2.
NOTE: The data set WORK.TEST has 3 observations and 4 variables.

Output:

<table>
<thead>
<tr>
<th>Obs</th>
<th>name</th>
<th>age</th>
<th>class</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jane</td>
<td>9</td>
<td>3rd grade</td>
<td>9:00am</td>
</tr>
<tr>
<td>2</td>
<td>Jane</td>
<td>10</td>
<td>5th grade</td>
<td>8:45am</td>
</tr>
<tr>
<td>3</td>
<td>Amy</td>
<td>12</td>
<td></td>
<td>8:30am</td>
</tr>
</tbody>
</table>

It is obvious that the above Output is incorrect, but the log did not indicate this error.

If OPTIONS MERGENOBY= WARN is used, the following warning message in the log will appear:
WARNING: No BY statement was specified for a MERGE statement.
NOTE: There were 2 observations read from the data set WORK.TEST1.
NOTE: There were 3 observations read from the data set WORK.TEST2.
NOTE: The data set WORK.TEST has 3 observations and 4 variables.

If OPTIONS MERGENOBY= ERROR is used, the following error and warning messages in the log will be clearer:
ERROR: No BY statement was specified for a MERGE statement.
NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.TEST may be incomplete. When this step was stopped there were 0 observation and 4 variables.

• OPTIONS REPLACE | NOREPLACE;
  REPLACE tells SAS that a permanently stored SAS data set can be overwritten with another SAS data set of the same name; this is the default.
  NOREPLACE is a useful option; it can prevent the accidental replacement of existing SAS data sets. A warning message will be sent stating the data set was not replaced because of NOREPLACE option. This option has no effect on data sets in the WORK library.

CONCLUSION

This paper lists a handful of System Options by the functionality group which help to make a programmer's job easier and more productive. Hopefully the information contained in this paper will result in SAS users paying more attention to this frequently underexplored area of SAS.
ACKNOWLEDGMENT

We want to thank Maryann Williams, Richard Lowry and other colleagues for reviewing this paper and their helpful suggestions.

CONTACT INFORMATION

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

Wei (Lisa) Lin
Merck & Co., Inc.
UG1CD-14
Po Box 1000
North Wales, PA 19454-1099
Work Phone: 267-305-6521
Email: lisa_lin@merck.com

Jiannan (Jane) Kang
Merck & Co., Inc.
UG1CD-14
PO Box 1000
North Wales, PA 19454-1099
Work Phone: 267-305-7426
Email: jiannan_kang@merck.com

TRADEMARKS

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.