I. INTRODUCTION

This tutorial discusses the functions of the SAS Supervisor during the execution of a SAS DATA Step program, and will be presented in both the Tutorial and the Advanced Tutorial sessions. A more detailed discussion will be given in the Advanced Tutorial session. The functions of the SAS Supervisor can be categorized as follows:

- Compiling SAS Source Code, and
- Executing Resultant Machine Code

The actions of the Supervisor during both the compile and execution phases of a SAS job will be illustrated.

When a SAS DATA Step program is written, the DATA Step "module" must be integrated within the structure of the SAS System. This integration is done by the SAS Supervisor. Gaining a more complete understanding of what the Supervisor does and how our "program" is controlled by it is crucial to using the SAS System more effectively.

II. STRUCTURE OF SAS JOBS

There are distinct compile and execute steps for all SAS jobs. This fact is not readily apparent since a single program, the SAS Supervisor, handles the compile and execution (including linkage-editing) steps of a SAS job. There is a distinct compile step and execution step for each DATA or PROC step in a SAS job. The DATA and PROC steps are compiled and executed independently according to their sequence in the program. In particular, the first DATA/PROC step is compiled and then executed; this is then followed by the compilation and the execution for the next DATA/PROC step, etc. The SAS Supervisor controls this processing.

The SAS programmer has tools that allows him or her to take full advantage of the compile/execute structure for SAS jobs. For example, through the use of the Macro Language, the programmer has control over the sequence of DATA/PROC steps seen by the Supervisor and of the statements contained within each step. There are other tools and techniques which are available to exercise control over Supervisor functions within a given DATA Step, such as conditional execution of a read operation, or reading data within a loop. The following sections discuss the actions of the Supervisor during compilation and execution of a DATA Step, and the coding techniques that can be used to control or override the Supervisor's default actions.

III. COMPILATION PROCESSING

During the compilation of a DATA Step, the Supervisor creates both permanent and transient (in that they "disappear" after the compilation or execution of the current DATA Step) entities. The primary permanent entity is the directory or header portion of the SAS data set (the data is added to the data set at execution time). The transient entities include a variety of buffers, flags and work areas which, at execution time, control the creation of the desired output. The following is a partial list of the more important actions taken by the SAS Supervisor during the compilation of a DATA Step:

- Syntax scan;
- Translation from SAS source code to machine language object code;
- Definition of input and output files including variable names, their locations and attributes;
- Creation of the Program Data Vector;
- Specification of variables to be written to the output SAS data set;
- Specification of variables which are to be initialized to missing by the SAS Supervisor between executions of the DATA Step and during read operations; and
- Creation of a variety of "flag variables" which are used by the Supervisor at execution time.

The last four actions in the above list will be discussed in the following subsections.

A. Creation of the Program Data Vector

The Program Data Vector (PDV) is a buffer which includes all variables referenced either explicitly or implicitly in the DATA Step. It is used
at execution time as the location where the working values of variables are stored as they are processed by the DATA Step "program." The PDV is created at compile time by the SAS Supervisor.

Variables are added to the PDV sequentially as they are encountered during parsing and interpretation of SAS source statements. The following rules are used in defining the variables and their attributes to the PDV:

1. A variable is added to the PDV by its first occurrence (explicit or implicit) in the SAS source statements.

2. DROP and KEEP statements and output data set name parameters are ignored for the purposes of adding variables to the PDV.

3. The SAS automatic variables (e.g., _N_, _ERROR_) are always added. They are added as they are referenced or created in the DATA Step program.

4. Variables can be implicitly referenced and thus added to the PDV through SET, MERGE or UPDATE statements. Variables referenced in this way are added to the PDV when the "read" statement is encountered at compile time, regardless of whether or not the statement is ever executed. The DROP and KEEP data set name parameters used on an input data set affect which variables are added to the PDV from that data set.

The use of these rules is illustrated for a sample program in Figure 1.

B. Specification of Variables for Output

The specification of the list of variables to be copied from the PDV to the output SAS data set is best illustrated by a buffer called the DROP/KEEP Table (DKT) that has a one-to-one relationship to the PDV in that it contains a column for each variable in the PDV. It contains a row for each output data set. However, unlike the PDV, the elements of the DKT can only take the values of "D" or "K," for drop and keep. Furthermore, its values are supplied at compile time and can not be altered during the execution phase of a DATA Step. The Supervisor uses the following rules in setting DKT values:

1. For each variable in the PDV, set all of its DKT values to "D" if it is a SAS special variable (e.g., _N_, _ERROR_, END=, IN=, POINT=, FIRST, and LAST variables, and implicit ARRAY indices). Otherwise set the DKT values to "K."

2. DROP statement changes to DKT are made before KEEP statement changes.

3. For each variable in a DROP statement with its DKT value equal to "K," change it to "D." If the DROPPED variable is not found, set an error condition. The error message is:

THE VARIABLE <variable name> IN THE DROP LIST HAS NEVER BEEN REFERENCED.

This message most often occurs when: a variable is listed in more than one DROP statement or more than once in a single DROP statement; a variable not in the PDV at all is listed in a DROP statement; or a SAS automatic variable is listed in a DROP statement.

4. If any KEEP statements are present, then create a list of unique variable names from all KEEP statements. The Supervisor compares this list with variables in the PDV that have their DKT value equal to "K." Matches between these two will have no change made to their DKT values. Process the mismatches as follows: for variables in the PDV with DKT equal to "K" but not in the list of unique variables from all KEEP statements, set the DKT value to "D"; for variables in the list compiled from all KEEP statements which do not match variables in the PDV with DKT equal to "K", set an error condition. The error message is:

THE VARIABLE <variable name> IN THE KEEP LIST HAS NEVER BEEN REFERENCED.

This message most often occurs when: a variable is listed in a DROP statement and a KEEP statement; a variable which is not in the PDV at all is listed in a KEEP statement; or a SAS automatic variable is listed in a KEEP statement.

5. For each output data set with DROP or KEEP data set name parameters, change its row in the DKT as follows:

5.1 Process DROP before KEEP.

5.2 For each variable listed in
DATA SALES (DROP=DISCOUNT SALESAMT) 
  DETAIL (DROP=TOTALSALES TOTDSCNT); 
  ------RETAIN TOTALSALES TOTDSCNT; 
  DROP UNITPRICE QUANTITY; 
  ------SET TRANSACTION (DROP=TRANSDATE); 
  BY IDNO TYPE; 
  ------SALESAMT=UNITPRICE*QUANTITY; 
  IF QUANTITY > 10 THEN DISCOUNT=ROUND(SALESAMT*.1,.01); 
  ELSE IF QUANTITY > 5 THEN DISCOUNT=ROUND(SALESAMT*.05,.01); 
  OUTPUT DETAIL; 
  TOTALSALES+SALESAMT; 
  TOTDSCNT+DISCOUNT; 
  IF LAST.TYPE THEN 
  DO; 
  OUTPUT SALES; 
  TOTALSALES=0; 
  TOTDSCNT=0; 
  END; 
  RUN;

NOTE: _N_, _ERROR_, FIRST.IDNO, LAST.IDNO and FIRST.TYPE are in the PDV. They 
are not included in the figure since they are not referenced by the program.

FIGURE 1. The Program Data Vector (PDV)

SALES

K K K K K D D D D D D

DETAIL

D D K K K D D K D

TOTALSALES TOTDSCNT IDNO TYPE UNITPRICE QUANTITY SALESAMT DISCOUNT LAST.TYPE

Figure 2. The DROP/KEEP Table (DKT)

N N N N N N N N Y Y Y N

TOTALSALES TOTDSCNT IDNO TYPE UNITPRICE QUANTITY SALESAMT DISCOUNT LAST.TYPE

Figure 3. The Initialize To Missing Vector (ITMV)
the DROP list, set its DKT value to 'D'.

5.3 For each variable not listed in the KEEP list, set its DKT value to 'D'.

5.4 Ignore variables in the DROP or KEEP list that are not found in the PDV.

The use of these rules is illustrated in Figure 2.

C. Initialization to Missing Values

The specification of the variables that are to be initialized to missing between every execution of the DATA Step program by the SAS Supervisor is also illustrated by a buffer with a one-to-one correspondence to the PDV. The elements of this Initialize To Missing Vector (ITMV) can take three possible values:

- Y means initialize to missing between each execution of the DATA Step.
- N means do not initialize to missing.
- R means that the read operation (i.e., SET, MERGE or UPDATE) will perform the initialization to missing values. This value is only used when multiple data sets are being read.

These values, like the values in the DKT, are defined at compile time and can not be changed at execution time. The ITMV values for all variables are initially set to "Y" and are changed to "N" for the following situations:

1. All SAS special variables.
2. All variables listed in a RETAIN statement (note that "RETAIN;" forces all variables to have ITMV set to "N").
3. All variables which are physically present as the accumulator variable (the variable to the left of the "+" sign) in a sum statement. An exception is when the variable name in the accumulator position for a sum statement is an ARRAY name. ITMV values for the ARRAY elements are not affected by this rule.

All variables which are referenced in SET, MERGE or UPDATE statements will have ITMV values set to "N", or "R" according to the following rules:

1. ITMV values are set to "N" for variables read from a single SAS data set with the SET statement.
2. Where two or more data sets are read with a SET, MERGE or UPDATE statement, ITMV values for the variables from those data sets are set to "R".

These rules are illustrated in Figure 3.

D. Process Control Flags

In addition to the above buffers or vectors, other flag variables are created during the compile phase of a DATA Step program. The Data Step Failed Flag (DSFF) and the End Data Step Flag (EDSF) are created at compile time; their values are supplied at execution time. The Output Statement Present Flag (OSPF) is created and its value is supplied at compile time. OSPF is set to "Y" if there is any output statement present in the DATA Step program, otherwise it is set to "N." The DSFF, EDSF and OSPF are all used by the SAS Supervisor during the execution phase to control DATA Step processing. Values for the PDV, DKT, ITMV, DSFF, EDSF and OSPF for a sample program at the completion of the compile phase are illustrated in Figure 4.

E. Other Compile Time Operations

It should be noted that the above represents only a subset of the SAS Supervisor compile time functions. All of the following statements ("non-executable" or "information" statements) do all of their work at compile time:

- ARRAY
- ATTRIB
- BY
- DROP
- FORMAT/INFORMAT
- KEEP
- LABEL
- LENGTH
- RENAME
- RETAIN

Because these statements have their effect at compile time, their location within the DATA Step code is irrelevant; they may be placed at the beginning, at the end, or anywhere within the DATA Step program. An exception to this is the LENGTH statement, which should always be placed at the beginning of the DATA Step. This ensures that variables are added to the PDV by the reference on the LENGTH statement. These points should be kept in mind when writing and debugging SAS programs.

IV. EXECUTION TIME ACTIVITIES

Once the DATA Step has been successfully compiled and all of the above described buffers and flags have
**THE PDV**

| 0 | 0 | . | . | . | . | . | 1 |

| TOTSALES | TOTDSCNT | IDNO | TYPE | UNITPRCE | QUANTITY | SALESAMT | DISCOUNT | LAST.TYPE |

**THE DKT**

**SALES**

| K | K | K | K | D | D | D | D | D |

| DETAIL |

| D | D | K | K | D | D | K | K | D |

| TOTSALES | TOTDSCNT | IDNO | TYPE | UNITPRCE | QUANTITY | SALESAMT | DISCOUNT | LAST.TYPE |

**THE ITMV**

| N | N | N | N | N | N | N | Y | Y | N |

| TOTSALES | TOTDSCNT | IDNO | TYPE | UNITPRCE | QUANTITY | SALESAMT | DISCOUNT | LAST.TYPE |

**Data Step Failed Flag (DSFF) = ""**

**End Data Step Flag (EDSF) = ""**

**Output Statement Present Flag (OSPF) = "Y"**

Figure 4. Summary of Compile Time Buffers and Flags

---

**FIGURE 5. Execution Time Flow**

---

34
been created, the execution phase of the DATA Step can begin. This is illustrated by the simple program flow in Figure 5. The SAS DATA Step can be viewed as a subroutine which is executed repeatedly by the SAS Supervisor, usually until there is no more input data. In a typical SAS job, the Supervisor does the following:

1. Initialization of variables in the PDV to missing.
2. Execution ("calling") of the DATA Step program.
3. Outputting or copying values of variables in the PDV to the output SAS data set.
4. Repeating steps 1-3 until the input data source is exhausted.

The details of what happens during the execution of the DATA Step program (step 2 above) is controlled by the user in their SAS code. The details of how the Supervisor performs steps 1,3 and 4, as described above, will be discussed in this section along with a description of how the buffers and flags (created during the compile phase) are used. It should be remembered that the actions of the SAS Supervisor within the execution phase of a DATA Step are geared towards one goal: the repeated execution of a DATA Step program. In other words, the DATA Step program can be viewed as the inside of a read-write loop.

A. Execution Time Program Flow

The SAS Supervisor performs initialization before every execution of our DATA Step program using the PDV and the ITMV as follows:

For each variable in the PDV with its corresponding ITMV = "Y," the SAS Supervisor will set its position in the PDV to missing ('.' for numeric, ' ' for character).

The DATA Step program is then executed (called). The programming statements that comprise the DATA Step are executed, supplying values for the variables in the PDV.

Once the DATA Step program has finished, control is returned to the SAS Supervisor which decides whether to copy the contents of the PDV to the output SAS data set. The OSPF, DSFF, DKT and PDV are used to do this as follows:

If OSPF = "N" and DSFF = "N" then execute the OUTPUT routine.

The OUTPUT routine, which is also invoked when an OUTPUT statement is executed from within the DATA Step program, can be described as follows:

For each variable in the PDV with its corresponding DKT = "K", copy its current value from the PDV to the output SAS data set.

The value for the OSPF is set at compile time. Values for DSFF and EDSF are set at execution time. The setting of these flags is discussed in the following paragraphs which also addresses the looping or repeated execution of the DATA Step program done by the SAS Supervisor.

On referring to Figure 5, the question arises as to how the SAS Supervisor knows when to stop executing the DATA Step program. The more detailed flow diagram given in Figure 6 is a more accurate representation of execution time processing, which can be described as follows:

1. During the INITIALIZATION phase, set the values of DSFF and EDSF to "N".
2. Execute the DATA Step program, statement by statement.

2.1 When executing the read operation (for this "generic" case, assume a simple SET or INPUT statement) call a Supervisor routine to:

2.1.1 Determine if there is more input data.

2.1.2 If no more data, set DSFF and EDSF to "Y" and skip the rest of the DATA Step program, returning control to the SAS Supervisor.

2.1.3 Otherwise, copy the variables from the input data set to the PDV, set the values of any appropriate special variables and return control to the next executable DATA Step statement immediately following the read operation statement. The SAS Supervisor resumes control after the last executable statement in the DATA Step has been executed.

3. If OSPF = "N" and DSFF = "N" then execute the OUTPUT Routine.
FIGURE 6. Detailed Execution Time Flow
4. If EDSF="Y" then end the DATA Step and proceed to the next DATA or PROC step. Otherwise, repeat the above steps.

In writing a SAS DATA Step program, it is crucial to keep in mind the statements that return control to the Supervisor and how they impact the values of the DSFF and EDSF flags. Execution of the following statements all cause an immediate return to the SAS Supervisor with the indicated values for the flags:

<table>
<thead>
<tr>
<th>STATEMENT</th>
<th>DSFF</th>
<th>EDSF</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABORT</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>DELETE</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>IF false &lt;expression&gt;</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>RETURN</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>STOP</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Failed read operation (i.e. INPUT, SET, MERGE or UPDATE)</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

On reviewing the above table and the rules for the SAS Supervisor OUTPUT, it is clear that when OSPF="Y", DELETE, a false subsetting IF <expression> and RETURN are equivalent, since the default Supervisor OUTPUT is dependent on OSPF="N" and DSFF="N". When OSPF="Y", the value of DSFF has no impact on the Supervisor's default OUTPUT. This is not readily apparent without an understanding of how the Supervisor works during DATA Step execution.

B. Read Operation Details

The SAS read operations SET and MERGE perform two general actions when executed:

- Call a SAS Supervisor routine to initialize selected variables in the PDV to missing.
- Copy variable values from one or more SAS data sets to the PDV.

These actions are performed according to a set of rules, depending on which type of read operation is being performed and whether or not a BY statement is present. The rules for each type of statement are examined below.

When a SET statement references more than one SAS data set, and no BY statement is present, the data sets listed on the SET statement are concatenated. The SET statement performs the following actions when executed:

1. Determine which data set is being read and set IN= and END= variable values.

2. If the SET statement will read from a different data set compared to its last execution, then initialize all variables in the PDV with ITMV values of "R" to missing.

3. Copy the values of variables from the current data set to the PDV.

When a SET statement referencing more than one SAS data set has a BY statement associated with it, the data sets listed on the SET statement are interleaved. The SET statement performs the following actions when executed:

1. Determine which data set is being read by looking ahead to the values of the variables in the BY statement for the next observation in each data set. Set values for IN= and END= variables.

2. If the observation to be read is the first observation for a new BY group, then do the following:
   2.1 Set the appropriate FIRST. variables to 1.
   2.2 Set all variables in the PDV with ITMV values of "R" to missing.

3. If the SET statement will read from a different data set compared to its last execution, regardless of whether the BY group changes, then initialize all variables in the PDV with ITMV values of "R" to missing.

4. Copy variable values to the PDV from the current data set.

5. Look ahead to the values of the variables in the BY statement for the next observation in each data set. If there are no more observations for this BY group then set the appropriate LAST. variables to 1.

When a MERGE statement with no BY statement is present, the observations in the data sets listed on the MERGE statement are merged one-to-one. The MERGE statement performs the following actions when executed:

1. Copy variables values to the PDV from next observation in the first data set listed on the MERGE statement, then the second data set, and so on until all data sets have been read.

2. If end-of-file has been reached for a data set and no observation is read, initialize variables unique to that data set to missing.
3. Set IN= variables depending on which data sets are read.

When a MERGE statement with a BY statement is executed, the observations in the data sets listed are merged according to the values of the variables on the BY statement. The MERGE statement performs the following actions when executed:

1. Determine which data sets are being read by looking ahead to the values of the variables in the BY statement for the next observation in each data set.

2. If the observation(s) to be read represent a new BY group, then do the following:
   2.1 Set the appropriate FIRST. variables to 1.
   2.2 Set all of the IN= variables to 0.
   2.3 Set all variables with ITMV values of "R" to missing.

3. For each data set listed on the MERGE statement having another observation for this BY group, do the following:
   3.1 Set the appropriate IN= variable to 1.
   3.2 Copy variable values from the data set to the PDV.

4. Look ahead to the next observation in each data set to determine if any more observations are present for this BY group. If not, set the appropriate LAST. variable values to 1.

Understanding these rules can be helpful when writing or debugging a SAS program with complex DATA Step code. The following sections discuss how the SAS programmer can take control from the SAS Supervisor within a DATA Step, and how the rules governing the actions of the Supervisor and of the read operations apply under those circumstances.

C. Reading SAS Data Sets Within a Loop

SAS programmers can take control from the SAS Supervisor within a DATA Step by placing the read operation statement inside a DO loop. When the SAS Supervisor does the looping, the read operation executes once for each execution of the DATA Step. There is some "overhead" involved in returning control to the Supervisor each time. Therefore, the programmer might consider reading large data sets within a DO loop in order to increase the efficiency of the DATA Step. There might be other reasons for reading data within a loop, such as when a data set must be searched to find a specific observation.

The program shown in Figure 7 merges two SAS data sets in one execution of the DATA Step. Because control is not returned to the Supervisor each time, an OUTPUT statement must be present inside the loop. If no OUTPUT statement were present, the output SAS data set within have only one observation. It is also important to remember that the value of _N_ is only incremented when the Supervisor begins a new execution of the DATA Step, thus the value of _N_ in the PDV will be equal to 1 for every observation processed.

Looping also has an effect on how variables in the PDV are initialized to missing. Variables with ITMV values of "Y" are only initialized to missing when program control is returned to the SAS Supervisor. These variables will NOT be initialized to missing as long as the loop is executing. Note that looping has no effect on the initialization of variables with ITMV values of "R" or "N".

It is important to remember which statements automatically return control to the Supervisor when executed. These statements should not be placed within the DO loop where the data is being read, since their execution will cause an exit from the DO loop.

D. Conditional Execution Of A Read Operation

Any SAS read operation can be executed conditionally. For example, suppose the programmer has a SAS data set with a single observation that contains a constant. This constant value is needed for each execution of the DATA Step. The SAS data set can be read by executing the SET statement only once, as shown in Figure 8. Since variables read from a SET statement referencing a single SAS data set have ITMV values set to "H", the constant will not be initialized to missing on subsequent executions of the DATA Step, therefore a RETAIN statement listing the variables in the data set OVERALL is not necessary.

E. Referencing A Data Set At Compile Time

Any SAS data set may be referenced at compile time only. For example,
suppose the programmer wants to add variables to the PDV, even though those variables will not be read in this DATA Step. This can be accomplished by "executing" the read operation based on a condition that will never be true. For example, at compile time the statement "IF 0 THEN SET INVDESC;" adds the variables in INVDESC to the PDV. At execution time no data will ever be read since 0 is false. This can be a useful technique for creating a "shell" of a SAS data set, where values are to be added later using FSEDIT or UPDATE.

Using the SET statement at compile time only can also be used in conjunction with the NOBS and POINT options. The program in Figure 9 sets a macro variable whose value is the number of observations in data set INVDESC. The SET statement is never executed because the "IF 0" condition is never true. However, the value for the NOBS= variable (N_OBS) is supplied at compile time. The only executable statements in the DATA Step are "CALL SYMPUT" and "STOP."

V. CONCLUSION

These examples illustrate that by gaining a more complete understanding of what the SAS Supervisor is doing at both compile and execution time, SAS programmers can make more informed decisions as to what they can let the SAS Supervisor do and what they should control themselves. This understanding should also permit the development of more flexible and efficient SAS programs.

The authors can be contacted at the following address:

Software Applications and Training Division
ORI, Inc.
Suite 1000
601 Indiana Avenue, N.W.
Washington, D.C. 20004
Phone: (202) 737-2666

**FIGURE 7.** The MERGE Statement Inside a Loop

```sas
DATA TOSELL;
  DO UNTIL(LASTREC);
    MERGE INVSALES INVDESC END=LASTREC;
    BY IDNO;
    OUTPUT;
  END;
  STOP;

PROC MEANS DATA=SALES NOPRINT;
  VAR TOTSALES;
  OUTPUT OUT=OVERALL SUM=ALLSALES;

DATA PERCENTS;
  SET SALES;
  IF _N_=1 THEN SET OVERALL;
  PERCENT=100*TOTSALES/ALLSALES;

FIGURE 8. Conditional Execution of a Read Operation

```