This tutorial discusses the function of the SAS Supervisor in the process of executing a SAS program (particularly DATA steps). An earlier tutorial, "Comparing SAS with Traditional programming Languages," discussed why and how SAS is different from languages such as Fortran, PL/I and COBOL. In the earlier tutorial, particular emphasis was given to the SAS "programming environment" which is a direct result of the existence and functions of the SAS Supervisor. This tutorial will expand on the functions of the SAS Supervisor: the major ones can be categorized as follows:

- Compiling SAS Source Code
- Executing Resultant Machine Code

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

When we write a SAS program, we are in fact writing a "module" which must be integrated with the SAS System. This integration is done by the SAS Supervisor (a program itself). Gaining a more complete understanding of what the Supervisor does and how our "program" is controlled by it is crucial to using SAS more effectively.

### Compile Phase

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

During the compile of a DATA step, the Supervisor creates both permanent and transient (in that they "disappear" after the 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 will be used at execution time in order to create the SAS data set. The following is a subset of the more important actions undertaken by the SAS Supervisor during the compilation of a DATA step:

- Syntax scan
- Translation to machine language object code from SAS source code
- Definition of input and output files including variable names, their locations and their attributes
- Creation of the Program Data Vector (PDV)
- Specification of variables to be written to the SAS data set
- Specification of variables which are to be initialized to missing by the SAS Supervisor between executions of the DATA step
- Creation of specific "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 paragraphs through the same program listed in Figure 1.

```sas
DATA PDNUS;
RETAIN TOTBNUS 0;
DROP ID R111E REVENUE BONUS;
SET SAlLES;
BY DEPT;
IF REVENUE GT 100000 THEN RATE=.1;
ELSE IF REVENUE GT 50000 THEN RATE=.05;
PDNUS=SLWlOOJ, RA1E*REVENUE);
TOIBJNUS=TOIBJNUS+PDNUS;
IF LAST .DEPT 111EN
W;
OUTPUT;
T01llONUS=D;
END:
```

**FIGURE 1. The sample program**
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 the parsing and interpretation of the 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 are ignored for the purposes of adding variables to the PDV. (Their use/function is discussed later.)
3. The SAS automatic variables _N_ and _ERROR_ are always added.
4. Other SAS special/automatic variables are added as they are referenced or created. Examples would be IN and END variables for a SET statement.
5. Variables can be implicitly referenced and thus added to the PDV through SET, MERGE or UPDATE statements.

The use of these rules is illustrated in Figure 2 for the sample program.

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 Vector) that bears a 1-1 correspondence to the PDV in that it contains a location for each variable in the PDV. However, unlike the PDV, the elements of the DROP/KEEP Vector (DKV), can only take the values "D" or "K." Further, its values are filled in at compile time and can not be altered during the execution phase. The following rules are used in setting the values for the DKV:

DATA BONUS;
  RETAIN TOTBONUS 0;
  DROP ID RATE REVENUE BONUS;
  SET SALES;
  BY DEPT;
  IF REVENUE GT 100000 THEN RATE=-1;
  ELSE IF REVENUE GT 50000 THEN RATE=-.05;
  BONUS=SUM(10000,RATE*REVENUE);
  TOTBONUS=TOTBONUS+BONUS;
  IF LAST.DEPT THEN DO;
  OUTPUT;
  TOTBONUS=0;
  END;

NOTE: The special/automatic variables _N_, _ERROR_, and FIRST.DEPT are in the PDV. They are not included in the figure since they are not referenced by the program.

FIGURE 2. The Program Data Vector
1. For each variable in the PDV, set its value in the OKV to "0" if it is a SAS special variable (e.g., _N_, _ERROR_, END=, IN=, FIRST, and LAST, variables). Otherwise set its value to "K".

2. Process all DROP statements before KEEP statements.

3. For each variable in a DROP statement with its DKV value equal to "K", change its DKV value to "D". If the DROPped variable is not found, set an error condition.

4. If a KEEP statement is present, then for each variable in the PDV with its DKV value equal to "K" but which is not listed in the KEEP statement, set its DKV value to "D". If the variable list in the KEEP statement is not exhausted by this process, set an error condition.

The use of these rules is illustrated in Figure 3.

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 best illustrated by a buffer with a 1-1 correspondence to the PDV. The elements of this Initialize To Missing Vector (ITMV) can only take the values "Y" or "N". These values, like the values in the DKV, are defined at compile time. The ITMV values for all variables start out as "Y" and are changed to "N" for:

1. All SAS special variables.

2. All variables listed in a RETAIN statement.

3. All variables which are physically present as the accumulator variable in a sum statement. If the "variable name" in the accumulator position for a sum statement is an ARRAY name, the ITMV values for the ARRAY's elements are unaffected by this rule.

4. All variables which may be read from an existing SAS data set by a SET, MERGE or UPDATE statement.

These rules may be summarized by stating that all variables created by DATA step programming statements have their ITMV flag set to "Y" with the exception of "sum" variables and variables listed in RETAIN statements. All other variables have their ITMV flag set to "N". These rules are illustrated in Figure 4.

DATA BONUS;
RETAIN TOTBONUS 0;
DROP ID RATE REVENUE BONUS;
SET SALES;
BY DEPT;
IF REVENUE GT 100000 THEN RATE=.1;
ELSE IF REVENUE GT 50000 THEN RATE=.05;
BONUS=SUM(.100000,RATE*REVENUE);
TOTBONUS=TOTBONUS+BONUS;
IF LAST.DEPT THEN
DO;
OUTPUT;
TOTBONUS=0;
END;

<table>
<thead>
<tr>
<th>&quot;K&quot;</th>
<th>&quot;N&quot;</th>
<th>&quot;D&quot;</th>
<th>&quot;D&quot;</th>
<th>&quot;N&quot;</th>
<th>&quot;N&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTBONUS</td>
<td>DEPT</td>
<td>ID</td>
<td>REVENUE</td>
<td>RATE</td>
<td>BONUS</td>
</tr>
</tbody>
</table>

FIGURE 3. The DROP/KEEP Vector
In addition to the above buffers or vectors, other flag variables are created during the compile phase. The Data Step Failed Flag (DSFF) and the End Data Step Flag (EDSF) are created at compile time, but their values are supplied at execution time. The Output Present Flag (OFF) is created and its value is defined at compile time. The OPF is set to "Y" if there is an OUTPUT statement present in the DATA step program, otherwise it is set to "N". All of the flags and vectors are illustrated for the sample program in Figure 5.

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

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

It is important to keep this point in mind when writing and debugging SAS programs.

DATA BONUS;
RETAIN TOTBONUS 0;
DROP ID RATE REVENUE BONUS;
SET SALES;
BY DEPT;
IF REVENUE GT 100000 THEN RATE=.1;
ELSE IF REVENUE GT 50000 THEN RATE=.05;
BONUS=SUM(1000, RATE*REVENUE);
TOTBONUS=TOTBONUS+BONUS;
IF LAST.DEPT THEN
DO;
OUTPUT;
TOTBONUS=0;
END:

<table>
<thead>
<tr>
<th>&quot;N&quot;</th>
<th>&quot;N&quot;</th>
<th>&quot;N&quot;</th>
<th>&quot;N&quot;</th>
<th>&quot;Y&quot;</th>
<th>&quot;Y&quot;</th>
<th>&quot;N&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTBONUS</td>
<td>DEPT</td>
<td>ID</td>
<td>REVENUE</td>
<td>RATE</td>
<td>BONUS</td>
<td>LAST.DEPT</td>
</tr>
</tbody>
</table>

Figure 4. The Initialize To Missing Vector

<table>
<thead>
<tr>
<th>&quot;N&quot;</th>
<th>&quot;N&quot;</th>
<th>&quot;N&quot;</th>
<th>&quot;N&quot;</th>
<th>&quot;Y&quot;</th>
<th>&quot;Y&quot;</th>
<th>&quot;N&quot;</th>
<th>&quot;N&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTBONUS</td>
<td>DEPT</td>
<td>ID</td>
<td>REVENUE</td>
<td>RATE</td>
<td>BONUS</td>
<td>LAST.DEPT</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5. Summary of compile time buffers and flags
EXECUTION PHASE

Once the DATA step has been successfully compiled and all of the above described buffers and tables have been created, the execution phase of the DATA step can begin. A graphic of this is given in Figure 6. The SAS code contained in the DATA step can be viewed as a program or subroutine which is executed repeatedly, usually until there is no more input data. This processing is controlled by the SAS Supervisor. In a "simple" SAS job, the SAS Supervisor performs the following during the execution phase:

- Initialization of selected variables in the PDV to missing
- Executing the DATA step program
- Outputting or copying values of selected variables in the PDV to the SAS data set
- Repeating the above three actions until the input data source is exhausted.

The details of what happens during the execution of the DATA step program is controlled by the user in their SAS code. The detailed processing that the SAS Supervisor uses to perform the other three items will be discussed in the following paragraphs. This discussion will describe how the buffers and flags, created during the compile phase, are used. It should be remembered however that the actions of the SAS Supervisor are geared towards one goal, the repeated execution of our DATA step program to create the desired output. Thus, we can view our DATA step program as the inside of a read-write loop.

The SAS Supervisor performs initialization before every execution of our DATA step program. The PDV and the ITMV are used to do this as follows:

For each variable (in the PDV) with its corresponding ITMV \( = "V" \), set its position in the PDV to missing ("." for numeric, "" for character).

The DATA step program is then executed (or called) to fill in values for the variables in the PDV.

Once the DATA step program has completed, control is returned to the SAS Supervisor which decides whether to copy the contents of the PDV to the SAS data set (output the observation). The Output Present Flag (OPF), Data Step Failed Flag (DSFF), DKV buffer and the PDV are used to do this as follows:

If \( \text{OPF} = "N" \) and \( \text{DSFF} = "N" \) 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 DKV = "K", copy its current value from the PDV to the output SAS data set.

Remember that a value for the OPF flag is set at compile time. Values for the DSFF flag and End Data Step Flag (EDSF) are set at execution time. How this is done is described in the following discussion of the looping or repeated execution of the DATA step program done by the SAS Supervisor.

On referring to Figure 6, the question arises as to how the SAS Supervisor knows when to stop executing the DATA step program. The flow diagram given in Figure 7 is a more detailed and accurate representation of execution time processing. It can be described as follows:

1. During the INITIALIZATION phase, set the values of DSFF and EDSF to "N".
2. Execute (call) the DATA step program.
3. The DATA step program will usually involve executing an INPUT, SET, MERGE or UPDATE statement (SET is used in the figure for the illustration) to read in data. This actually involves executing a Supervisor routine to:

3.1 Determine if there is more input data.

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

3.3 Otherwise, copy the variables from the input data set into the PDV, set the values of any appropriate special variables (e.g., IN, END, FIRST, or LAST variables) and return to the DATA step program statement immediately after the SET (or INPUT, MERGE, UPDATE).

4. If appropriate, the SAS Supervisor outputs the observation to the SAS data set.

5. If EDSF = "Y" then end the DATA step and proceed to the next DATA or PROC step. Otherwise, repeat the above processing 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. The following statements all cause an immediate return (i.e., the remaining DATA step programming statements are skipped) to the 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>&quot;Y&quot;</td>
<td>&quot;Y&quot;</td>
</tr>
<tr>
<td>DELETE</td>
<td>&quot;Y&quot;</td>
<td>&quot;N&quot;</td>
</tr>
<tr>
<td>Subsetting IF</td>
<td>&quot;Y&quot;</td>
<td>&quot;N&quot;</td>
</tr>
<tr>
<td>RETURN</td>
<td>&quot;N&quot;</td>
<td>&quot;N&quot;</td>
</tr>
<tr>
<td>Failed SET, MERGE, UPDATE or INPUT</td>
<td>&quot;Y&quot;</td>
<td>&quot;Y&quot;</td>
</tr>
<tr>
<td>STOP</td>
<td>&quot;N&quot;</td>
<td>&quot;N&quot;</td>
</tr>
</tbody>
</table>

On reviewing the above table and the rules for the SAS Supervisor OUTPUT, it becomes clear that when OPF = "Y" (i.e., an OUTPUT statement is present), DELETE, IF false and RETURN are equivalent. This is not a readily apparent fact without an understanding of how the SAS Supervisor works during the execution phase of a DATA step.

EXAMPLE AND SUMMARY

The above discussion will be summarized by discussing the sample program listed in Figure 1. This program is repeated in Figure 8. The right-hand column lists the SAS statements contained in the DATA step program. The left-hand column contains a description of what the SAS Supervisor is doing. Note that the DATA step program is doing all of the OUTPUTting and selected initialization (TOTBONUS = 0). The SAS Supervisor is doing some initialization (BONUS and DATA BONUS;)

\[
\text{DATA BONUS;}
\]

\[
\text{RETAIN TOTBONUS 0;}
\]

\[
\text{DROP ID RATE REVENUE BONUS;}
\]

\[
\text{SET SALES;}
\]

\[
\text{BY DEPT;}
\]

\[
\text{IF REVENUE GT 100000 THEN RATE=.1;}
\]

\[
\text{ELSE IF REVENUE GT 50000 THEN RATE=.05;}
\]

\[
\text{BONUS=SUM(10000,RATE*REVENUE);}
\]

\[
\text{TOTBONUS=TOTBONUS+BONUS;}
\]

\[
\text{IF LAST.DEPT THEN}
\]

\[
\text{DO;}
\]

\[
\text{OUTPUT;}
\]

\[
\text{TOTBONUS=0;}
\]

\[
\text{END;}
\]

FIGURE 8. Supervisor versus DATA step program

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. The following statements all cause an immediate return (i.e., the remaining DATA step programming statements are skipped) to the 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>&quot;Y&quot;</td>
<td>&quot;Y&quot;</td>
</tr>
<tr>
<td>DELETE</td>
<td>&quot;Y&quot;</td>
<td>&quot;N&quot;</td>
</tr>
<tr>
<td>Subsetting IF</td>
<td>&quot;Y&quot;</td>
<td>&quot;N&quot;</td>
</tr>
<tr>
<td>RETURN</td>
<td>&quot;N&quot;</td>
<td>&quot;N&quot;</td>
</tr>
<tr>
<td>Failed SET, MERGE, UPDATE or INPUT</td>
<td>&quot;Y&quot;</td>
<td>&quot;Y&quot;</td>
</tr>
<tr>
<td>STOP</td>
<td>&quot;N&quot;</td>
<td>&quot;N&quot;</td>
</tr>
</tbody>
</table>

On reviewing the above table and the rules for the SAS Supervisor OUTPUT, it becomes clear that when OPF = "Y" (i.e., an OUTPUT statement is present), DELETE, IF false and RETURN are equivalent. This is not a readily apparent fact without an understanding of how the SAS Supervisor works during the execution phase of a DATA step.

EXAMPLE AND SUMMARY

The above discussion will be summarized by discussing the sample program listed in Figure 1. This program is repeated in Figure 8. The right-hand column lists the SAS statements contained in the DATA step program. The left-hand column contains a description of what the SAS Supervisor is doing. Note that the DATA step program is doing all of the OUTPUTting and selected initialization (TOTBONUS = 0). The SAS Supervisor is doing some initialization (BONUS and DATA BONUS;)

\[
\text{DATA BONUS;}
\]

\[
\text{RETAIN TOTBONUS 0;}
\]

\[
\text{DROP ID RATE REVENUE BONUS;}
\]

\[
\text{SET SALES;}
\]

\[
\text{BY DEPT;}
\]

\[
\text{IF REVENUE GT 100000 THEN RATE=.1;}
\]

\[
\text{ELSE IF REVENUE GT 50000 THEN RATE=.05;}
\]

\[
\text{BONUS=SUM(10000,RATE*REVENUE);}
\]

\[
\text{TOTBONUS=TOTBONUS+BONUS;}
\]

\[
\text{IF LAST.DEPT THEN}
\]

\[
\text{DO;}
\]

\[
\text{OUTPUT;}
\]

\[
\text{TOTBONUS=0;}
\]

\[
\text{END;}
\]

FIGURE 8. Supervisor versus DATA step program

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. The following statements all cause an immediate return (i.e., the remaining DATA step programming statements are skipped) to the 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>&quot;Y&quot;</td>
<td>&quot;Y&quot;</td>
</tr>
<tr>
<td>DELETE</td>
<td>&quot;Y&quot;</td>
<td>&quot;N&quot;</td>
</tr>
<tr>
<td>Subsetting IF</td>
<td>&quot;Y&quot;</td>
<td>&quot;N&quot;</td>
</tr>
<tr>
<td>RETURN</td>
<td>&quot;N&quot;</td>
<td>&quot;N&quot;</td>
</tr>
<tr>
<td>Failed SET, MERGE, UPDATE or INPUT</td>
<td>&quot;Y&quot;</td>
<td>&quot;Y&quot;</td>
</tr>
<tr>
<td>STOP</td>
<td>&quot;N&quot;</td>
<td>&quot;N&quot;</td>
</tr>
</tbody>
</table>

On reviewing the above table and the rules for the SAS Supervisor OUTPUT, it becomes clear that when OPF = "Y" (i.e., an OUTPUT statement is present), DELETE, IF false and RETURN are equivalent. This is not a readily apparent fact without an understanding of how the SAS Supervisor works during the execution phase of a DATA step.

EXAMPLE AND SUMMARY

The above discussion will be summarized by discussing the sample program listed in Figure 1. This program is repeated in Figure 8. The right-hand column lists the SAS statements contained in the DATA step program. The left-hand column contains a description of what the SAS Supervisor is doing. Note that the DATA step program is doing all of the OUTPUTting and selected initialization (TOTBONUS = 0). The SAS Supervisor is doing some initialization (BONUS and DATA BONUS;)

\[
\text{DATA BONUS;}
\]

\[
\text{RETAIN TOTBONUS 0;}
\]

\[
\text{DROP ID RATE REVENUE BONUS;}
\]

\[
\text{SET SALES;}
\]

\[
\text{BY DEPT;}
\]

\[
\text{IF REVENUE GT 100000 THEN RATE=.1;}
\]

\[
\text{ELSE IF REVENUE GT 50000 THEN RATE=.05;}
\]

\[
\text{BONUS=SUM(10000,RATE*REVENUE);}
\]

\[
\text{TOTBONUS=TOTBONUS+BONUS;}
\]

\[
\text{IF \text{LAST.DEPT THEN}}
\]

\[
\text{DO;}
\]

\[
\text{OUTPUT;}
\]

\[
\text{TOTBONUS=0;}
\]

\[
\text{END;}
\]

FIGURE 8. Supervisor versus DATA step program

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. The following statements all cause an immediate return (i.e., the remaining DATA step programming statements are skipped) to the 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>&quot;Y&quot;</td>
<td>&quot;Y&quot;</td>
</tr>
<tr>
<td>DELETE</td>
<td>&quot;Y&quot;</td>
<td>&quot;N&quot;</td>
</tr>
<tr>
<td>Subsetting IF</td>
<td>&quot;Y&quot;</td>
<td>&quot;N&quot;</td>
</tr>
<tr>
<td>RETURN</td>
<td>&quot;N&quot;</td>
<td>&quot;N&quot;</td>
</tr>
<tr>
<td>Failed SET, MERGE, UPDATE or INPUT</td>
<td>&quot;Y&quot;</td>
<td>&quot;Y&quot;</td>
</tr>
<tr>
<td>STOP</td>
<td>&quot;N&quot;</td>
<td>&quot;N&quot;</td>
</tr>
</tbody>
</table>

On reviewing the above table and the rules for the SAS Supervisor OUTPUT, it becomes clear that when OPF = "Y" (i.e., an OUTPUT statement is present), DELETE, IF false and RETURN are equivalent. This is not a readily apparent fact without an understanding of how the SAS Supervisor works during the execution phase of a DATA step.

EXAMPLE AND SUMMARY

The above discussion will be summarized by discussing the sample program listed in Figure 1. This program is repeated in Figure 8. The right-hand column lists the SAS statements contained in the DATA step program. The left-hand column contains a description of what the SAS Supervisor is doing. Note that the DATA step program is doing all of the OUTPUTting and selected initialization (TOTBONUS = 0). The SAS Supervisor is doing some initialization (BONUS and DATA BONUS;)

\[
\text{DATA BONUS;}
\]

\[
\text{RETAIN TOTBONUS 0;}
\]

\[
\text{DROP ID RATE REVENUE BONUS;}
\]

\[
\text{SET SALES;}
\]

\[
\text{BY DEPT;}
\]

\[
\text{IF REVENUE GT 100000 THEN RATE=.1;}
\]

\[
\text{ELSE IF REVENUE GT 50000 THEN RATE=.05;}
\]

\[
\text{ELSE RATE=0;}
\]

\[
\text{BONUS=100000*RATExREVENUE;}
\]

\[
\text{TOTBONUS=SUM(TOTBONUS, BONUS);}
\]

\[
\text{END;}
\]

FIGURE 8. Supervisor versus DATA step program
RATE) and all of the looping. An equivalent program (in terms of its results) is listed in Figure 9. The program is repeated in Figure 10 with the program itself in the right-hand column and the SAS Supervisor actions in the left-hand column. Note that by moving some of the looping into the DATA step program, it is executed once for each DEPT. Thus we can let the SAS Supervisor do all of the initialization and the OUTPUTting.

This example illustrates 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 SAS do and what they should do themselves. This understanding should also permit the development of more flexible and efficient SAS programs.

The author can be contacted by calling 301-656-3276 or writing:

Don Henderson
Computer Services Division
ORI, Inc.
7910 Woodmont Avenue
Bethesda, MD 20814

DATA BONUS;

DO UNTIL(EDSF="Y")
  BONUS=,
  TOTBONUS=,
  RATE=,
  KEEP TOTBONUS DEPT;
  DO UNTIL(LAST.DEPT);
  COPY DEPT, ID,REVENUE TO PDV
  ASSIGN LAST.DEPT IN PDV
  ASSIGN DSFF AND EDSF
  IF EDSF="Y" RETURN

  IF REVENUE GT 100000 THEN RATE=.1;
  ELSE IF REVENUE GT 50000 THEN RATE=.05;
  ELSE RATE=0;
  BONUS=1000+RATE*REVENUE;
  TOTBONUS=SUM(TOTBONUS,BONUS);
END;

COPY TOTBONUS AND DEPT FROM PDV TO DATA SET BONUS
END

FIGURE 10. Supervisor versus "equivalent" DATA step program