PhUSE 2009

Paper TU04

Works as Designed (?)

Dr Heinrich von Lips, Ecron Acunova GmbH, Frankfurt am Main, Germany

ABSTRACT

When writing program code the SAS® programmer always has to consider how the code is interpreted by the SAS application during compilation and execution. Even though most SAS programmers are quite familiar with SAS data steps, he/she should be aware of possible pitfalls during the processing of SAS data steps. A key component during executing a SAS data step is the Program Data Vector (PDV). The elements of the PDV will be described considering different SAS data step statements and it will illustrate how the SAS data step is processed. This presentation will pick up a simple example that shows an unexpected result and provides a comprehensive explanation for the results.

INTRODUCTION

The title “Works as Designed (?)” might be rather provocative. However, even experienced SAS programmers come across the situation that the SAS program they have written does not deliver the desired or intended results. And the SAS application does not show any errors, warnings or appropriate notes during compilation and execution to indicate the unexpected results. It is obvious that the SAS application processes the data in a standard way. The SAS programmer must be aware of the processing to avoid unexpected results.

The Program Data Vector (PDV) is an element of the data processing of the SAS application. A simple SAS program concatenates and manipulates two SAS data sets. The result is not that one to be expected when following the logic of the SAS data step statements. The internal process of the SAS data steps in conjunction with the PDV will be explained. This will help to understand how the unexpected result of the given example comes about. Some rules for writing SAS programs are provided to help to avoid this kind of unexpected results.

The samples given here are generated with SAS 9.1.3, but the concept on the PDV applies also to other versions.

THE PROBLEM

The task is to concatenate two SAS data sets and change values of a variable depending on the value of another variable. For this purpose two data sets (`a_data` and `b_data`) are provided.
The data set `a_data` is shown in the following table.

Table1: SAS data set `a_data` with three variables and three observations.

ID	VAR1	VAR2
a1	1	11
a2	1	12
a3	2	23

The SAS data sets `b_data` looks similar and is given in the Table 2.

Table 2: SAS data set `b_data` with three variables and two observations.

ID	VAR1	VAR3
b1	1	101
b2	3	302

Each data set consists of three variables. The common variables for both data sets are ID and VAR1. The values of the variable ID are composed of the data set name and the row number. The ID variable will be used in the explanations for identification of the data. The third variables of the data sets `a_data` and `b_data` are VAR2 and VAR3, respectively.

In Program 1 the two data sets a_data and b_data are concatenated. In the same SAS data step the value of VAR3 is modified when the value of VAR1 is equal to 1. Also a new variable VAR4 is created which is set to 0 depending on the value of VAR1.

Program 1: Concatenation of two SAS data sets with data modifications.

```
DATA conc_data;
  SET a_data b_data;
  IF var1 EQ 1 THEN var3 = 0;
  IF var1 EQ 1 THEN var4 = 0;
RUN;
```

Looking at the input data sets and the SAS data step statements of Program 1 one would expect the result of Table 3.

Table 3: Expected content of the output data set conc_data generated by the SAS data step of Program 1.

ID	VAR1	VAR2	VAR3	VAR4
a1	1	11	0	0
a2	1	12	0	0
a3	2	23	.	.
b1	1	.	0	0
b2	3	.	302	.

The output data set conc_data consists of 5 observations as the SET statement concatenates the observations from both data sets a_data and b_data.
The variables ID and VAR1 contain the unchanged values from both input data sets.
As VAR2 is only present in data set a_data, VAR2 has just values for ID ‘a1’ to ‘a3’. For ID ‘b1’ and ‘b2’ the values of VAR2 are missing (represented by a period).
For VAR3 it is just the other way around. Only input data set b_data provides any data for VAR3. In addition, the SAS data step statement `IF var1 EQ 1 THEN var3 = 0` will change the value of VAR3 to 0 whenever the value of VAR1 is 1. Therefore, the value of VAR3 for ID ‘a1’, ‘a2’, and ‘b1’ is equal 0. For ID ‘b2’ the original value of the input data set b_data is given. For ID ‘a3’ a missing value is expected as no data is provided from input data set a_data.
VAR4 is newly generated within the SAS data step. The SAS data step statement `IF var1 EQ 1 THEN var4 = 0` sets the value of VAR4, just like for VAR3, to 0, whenever the value of VAR1 is 1. Otherwise the value of VAR4 is missing.

As mentioned above, this is what one would expect as the result of Program 1 just looking at the data sets and the program code without considering the internal data processing of the SAS application. However, the actual result is different. It is given in Table 4.

Table 4: Actual content of the output data set conc_data generated by the SAS data step of Program 1.

ID	VAR1	VAR2	VAR3	VAR4
a1	1	11	0	0
a2	1	12	0	0
a3	2	23	0	.
b1	1	.	0	0
b2	3	.	302	.

The difference between Table 3 and Table 4 can be seen for ID ‘a3’ where VAR3 has a value of 0, as VAR4 still has the expected missing value. As the IF statements to change the data for VAR3 and VAR4 are identical, there must be a reason which can not be found merely in the program code.

To understand this unexpected result it is necessary to understand how SAS processes data reading from SAS data sets.

THE PROGRAM DATA VECTOR

When using an INPUT, MERGE, SET, MODIFY, or UPDATE statement to read an observation from a SAS data set, the so-called Program Data Vector (PDV) is used for data processing.

DEFINITION

The Program Data Vector is a logical area in memory where SAS builds a data set, one observation at a time. When a program executes, SAS reads data values from the input buffer or creates them by executing SAS language statements. The data values are assigned to the appropriate variables in the Program Data Vector. From here, SAS writes the values to a SAS data set as a single observation.

This definition is taken from "SAS® 9.1.3 Language Reference: Concepts, Second Edition". However, the PDV is a general concept of the SAS application and is therefore not bound to a certain version.

The input buffer mentioned in the definition is also a logical area in memory used for reading data with an INPUT statement from external files or from DATALINES in a SAS data step. The definition leaves out the reading of data from SAS data sets. However, the concept of the PDV applies for reading from SAS data sets, too. Therefore, in the definition the term input buffer can be replaced by input data set.

COMPILATION

During compilation of the data step statements, the SAS application determines the structure of the PDV. The PDV contains all the variables from the input data set(s), the variables created by the data step statements and the additional internal variables automatically generated. The order of the variables in the program data vectors is determined by the time they are encountered.

The attributes of the variables in the PDV are the same as those in the input data set(s), if not changed in the data step statements. If the data step statements change the attributes e.g. FORMAT, LABEL, the new attributes are taken for the PDV. Variables created with the data step statements without any attributes explicitly assigned to have the default values given by SAS (e.g. length 8 for numeric variables).

In addition to the variables from the input data sets and the variables created in the SAS data step, automatic variables are created by data step statements or data set options. These automatic variables exist only temporarily and are used for internal processing purposes. They will not be written to the output data set.
Two automatic variables are always created. The variable _N_ represents the number of time the data step has iterated. The variable _ERROR_ is a flag with the values 0 for no error occurred during the execution of the SAS data step statement and 1 in case of one or more errors occurred (e.g. data type mismatch).
In addition further automatic variables exist depending on the data step statements or data set options. For example in case of a BY statement the variables FIRST.var and LAST.var are present in the PDV (var is representing the BY variable). In case of an IN data set option the assigned variable is part of the PDV. These automatic variables have the values 1 or 0 in case the incident is true or false, respectively.

Furthermore, a flag exists which indicates if a variable is dropped or kept after execution of the data step. E.g. automatic variables are always marked for dropping.

In Table 5 the structure of the PDV for the output data set conc_data after compilation of Program 1 is shown with all variables. The first column is only a description of the rows and is not part of the PDV.

Table 5: Structure of the PDV for the output data set conc_data after compilation of Program 1.

Variable name	ID	VAR1	VAR2	VAR3	VAR4	_N_	_ERROR_
Value		.	.	.	.	1	0
Flag	keep	keep	keep	keep	keep	drop	drop

The PDV contains the variables ID, VAR1, VAR2, VAR3, VAR4, _N_, and _ERROR_. Initially all values of the data set variables are set to missing (represented by blank for character and a period for numeric variables). The automatic variable _N_ is set to 1 (first iteration) and the automatic variable _ERROR_ is set to 0 (no error occurred).

EXECUTION

During execution, the data step statements are processed step by step for each observation read. The flow of a simple program (like Program 1) is described in the following:

1, The first observation is read from the input data set(s) and assigned to the corresponding variable in the PDV. Only one observation is present in the PDV at a given time.
2, Each data step statement is executed and the values of the variables are changed in case of an appropriate statement.
3, When all data step statements are executed, the actual content of the PDV is written to the output data set.
4, Then the automatic variable _N_ is iterated by one, the data step is processed again by reading the next observation from the input data set into the PDV and executing the subsequent data step statements.
5, The data step terminates when all observations are read from the input data sets.

This description of the execution applies of course to a simple SAS data step. But also for more complex SAS data steps the processing is done observation by observation. The PDV is the element where the data is collected and manipulated and from which the content is written to the output data set.

As the PDV is introduced and described now, the unexpected result can be explained.

THE EXPLANATION

So far the processing of data reading, manipulating, and writing was described. However what happens to the PDV after writing the data to the output data set and before reading the next observation into the PDV? Is the PDV totally cleared which means that all values of the PDV are set to missing?

CLEARANCE OF PDV

In general the PDV is not reset after writing the data to the output data set. This means that the old values of the previous observation remain in the PDV until they are overwritten by the next observation when it is read from the input data set.

However there are exceptions to this general rule. There are cases when the values of the variables are cleared by setting them to missing. (Note that clearance of the PDV will never affect the automatic variables, as their values are always updated at the beginning of an iteration.) These cases are:

1, At the beginning of the first iteration the PDV is still cleared. At that time no previous observation was read into the PDV.
2, If there is more than one data set in a `SET` statement the PDV is cleared after the last observation of the previous input data set and before reading the first observation of the next input data set.
3, If the `BY` statement is used the PDV is reset every time when the `BY` group changes. This means when the value of the `BY` variable changes, the PDV is cleared.
4, The variable is created in the SAS data step and does not originate from any input data set. Then the clearance occurs for each iteration.
5, Before data is read within a SAS data step from `DATALINES` the PDV is cleared.

To sum up we get the following rules:

1, Variables created by SAS data steps are ALWAYS cleared. (If there is an explicit `RETAIN` statement, even these variables of course are not reset.)
2, Variables from input data sets are cleared under certain conditions.
3, All automatic variables are NEVER reset, but always updated at the beginning of the iteration or at the `SET` statement.

This means that all values of the variables read from an input data set are retained until they are overwritten by a new value from the next observation (presuming none of the above described conditions for clearance apply). Note that a variable with a missing value in the input data set will also overwrite the old value in the PDV.

EXAMPLE OF CONCATINATION

Coming back to the example given in Program 1, the unexpected result for `conc_data` shown in Table 4 becomes clear now.

The PDV of conc_data given in Table 5 contains all variables from both input data set `a_data` and `b_data`. When reading the first input data set `a_data`, only the variables ID, VAR1, AND VAR2 are updated with the values of the read observation. The variable VAR3 however is not overwritten as the variable does not exist in the data set

b_data. The same is true when the variable VAR2 is updated with SAS data step statements while reading the second input data set b_data.

In the SAS data steps variable VAR3 is changed depending on the value of variable VAR1. However because VAR3 is not in the input data set a_data the value is not changed once a value other then missing is assigned to it. Therefore the value (in this case 0) is retained until the new input data set b_data is read. At the beginning of the input data set the values in the PDV are reset. (And it is also overwritten by the values of the input data set b_data.)

The variable VAR4 is always cleared at the beginning of each iteration as it does not exist in either of the input data sets but is created in the SAS data step. That is the reason why the results for VAR4 in Table 4 were exactly as the expected ones (see Table 3).

Program 2 demonstrates the clearance of the PDV when data is read from a new input data set. Program 2 is almost the same as Program 1 with the slight modification that EQ was replaced by NE in the IF statement and the order of the input data sets is changed. The content of conc_data is shown in Table 6.

Program 2: Same as Program 1 except that the values of VAR3 and VAR4 are set to 0 when VAR1 is not equal to 1. Also input data set b_data is read before input data set a_data.

```
DATA conc_data;
  SET b_data a_data;
  IF var1 NE 1 THEN var3 = 0;
  IF var1 NE 1 THEN var4 = 0;
RUN;
```

Table 6: Content of the output data set conc_data generated by the SAS data step of Program 2.

ID	VAR1	VAR3	VAR2	VAR4
b1	1	101	.	.
b2	3	0	.	0
a1	1	.	11	.
a2	1	.	12	.
a3	2	0	23	0

As seen in Table 6 the values of VAR3 and VAR4 are only 0 when the value of VAR1 is not equal to 1. The retaining of the value for VAR3 is stopped due to the clearance of the PDV before the first observation is read from the second input data set. This means that the actual and the expected results are only identical because the data manipulation was done on the last observation of the input data set b_data and not because this SAS program works better than Program 1.

Note that VAR3 is on the third position of the result table as the input data set b_data comes first in the SET statement. The structure of the PDV depends on the first occurrence of the variables.

EXAMPLE OF MERGE

But why is the PDV not always cleared? The reason for this concept becomes clear by looking at the merging of data sets. In Program 3 the two input data sets are merged by variable VAR1. This is a one to many relationship as data set a_data has two observations with VAR1 equal to 1 and data set b_data has only one observation.

Program 3: Merge of two SAS data sets a_data and b_data.

```
DATA merge_data;
  MERGE a_data b_data;
  BY var1;
RUN;
```

To illustrate the processing during merging the content of the PDV (not the output data set merge_data) after each iteration is given in Table 7.

Table 7: The content of the PDV after each iteration of Program 3. The observation numbers from the input data sets (last two columns) are added to the PDV for explanation.

ID	VAR1	VAR2	VAR3	FIRST. VAR1	LAST. VAR1	_ERROR_	_N_	Observation number from a	Observation number from b
b1	1	11	101	1	0	0	1	1	1
a2	1	12	101	0	1	0	2	2	1
a3	2	23	.	1	1	0	3	3	-
b2	3	.	302	1	1	0	4	-	2

As mentioned before, the PDV contains all variables from the input data set a_data and b_data as well as the standard automatic variables. As a BY statement is used in the SAS data step, the automatic variables FIRST.VAR1 and LAST.VAR1 are also present in the PDV. In Table 7 there are two additional columns that give the observation numbers of the input data sets to show how the data is combined. A dash in these columns means that no observation from this data set contributes to the output data set for the particular row. These columns are not part of the PDV.

When merging the data sets, first the data is read from input data set a_data into the PDV and then the data from input data set b_data is transferred. As ID is not part of the BY statement the value of ID from input data set b_data overwrites the value of input data set a_data in the PDV. That is why for _N_ = 1 the value of ID is 'b1'. On the next iteration data is only read from input data set a_data. The values of the first observation from input data set b_data are retained. Therefore the value of the second observation of input data set a_data overwrites the value of ID in the PDV (ID = 'a2').

As previously mentioned the PDV is always cleared when a new BY group starts. For _N_ = 3 and _N_ = 4 the value of VAR1 as the BY variable has changed and the PDV is cleared. Therefore, the values of VAR3 and VAR2 are missing for _N_ = 3 and _N_ = 4, respectively.

If the PDV was not cleared before the first observation of the next BY group was read, the value would be retained. E.g. for _N_ = 4 the PDV would contain values read from b_data with VAR1 = 3 and the retained value of VAR2 with VAR1 = 2. This, however, would contradict the merging principle when using BY statements.

If on the other hand the PDV was cleared after each iteration of the data step, the merging would not work without reading the values for the observation again. E.g. for _N_ = 2 the first observation from input data set b_data has to be re-read after reading the second observation of the input data set a_data. Re-reading data would make the entire data processing inefficient.

THE CONSEQUENCES

As the PDV is explained and the how the SAS application processes data is clear, the results of Program 1 given in Table 4 are not unexpected any more, but still undesired. What can be done to avoid these undesired results?

There are probably many possibilities to avoid the undesired results. Here are two examples.

In Program 4 the SAS data set option IN is used to indicate from which input data set the data is read. An ELSE statement and the condition whether the data is read from input data set a_data is added (ELSE IF novar3 THEN). VAR3 is set to missing when the data comes from input data set a_data. VAR3 is always read from input data set b_data overwriting the value in the PDV.

Program 4: Avoiding undesired results when setting two input data sets by using the data set option IN.

```
DATA conc_data;
  SET a_data (IN=novar3) b_data (IN=novar2);
  IF var1 EQ 1   THEN var3 = 0;
  ELSE IF novar3 THEN var3 = .;
  IF var1 EQ 1   THEN var4 = 0;
RUN;
```

This solution is straight forward but might become complex if more than two data sets or many variables are involved.

In Program 5 the setting of the two input data sets is done in one data step and the data manipulation is done in a second data step. Besides VAR4 which is created in the SAS data step and is always cleared for each new iteration, the PDV of conc_data2 contains the variables only from one data set (conc_data1). Therefore, the values in the PDV are always overwritten when reading the next observation.

Program 5: Avoiding undesired results when setting two input data sets by splitting the setting and the data manipulation in two data steps.

```
DATA conc_data1;
  SET a_data b_data;
RUN;

DATA conc_data2;
  SET conc_data1;
  IF var1 = 1 THEN var3 = 0;
  IF var1 = 1 THEN var4 = 0;
RUN;
```

Although this solution looks quite simple, this might not be a good approach for huge data. As the data has to be read twice it might be significantly more time-consuming. Maybe the UPDATE statement might help to reduce time.

CONCLUSION

Picking up the title of the presentation the PDV works as designed. If the SAS program works as designed depends on the SAS program. Being aware of the concept of the PDV and the data processing, undesired results can be avoided. The SAS programmer has to take control that the SAS application does what he/she wants.

It becomes clear form the given example that care has to be taken when reading data from several input data sets and manipulating data all within one SAS data step.

REFERENCES

SAS® 9.1.3 Language Reference: Concepts, Second Edition

CONTACT INFORMATION

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

Dr Heinrich von Lips
Ecron Acunova GmbH
Hahnstrasse 70
60528 Frankfurt am Main
Germany
Work Phone: +49 (69) 66 80 30 0
Fax: +49 (69) 66 80 30 29
Email: Heinrich.vonLips@ecronacunova.com
Web: www.ecronacunova.com

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.