

This Old Code: Tips for Renovating SAS® Release 5.18 Programs

Mary MacDougall, SPS Software Services, Inc., Canton, OH

ABSTRACT

Release 5.18 and prior releases for MVS are not Year 2000 compliant. If you have SAS Release 5.18 programs that must continue to run after December 31, 1999, you should convert them to use a later release of SAS, such as 6.09E or 7.

Most code written for SAS Release 5.18 can be run with newer releases of SAS. However, some programs will require modifications for the following reasons:

- Some options and keywords are no longer valid.
- Some non-standard syntax that was permitted in Release 5.18 could cause error conditions.
- There is a change in how SAS software processes certain calculations.
- Some default settings have changed.
- The appearance and content of some procedure output has changed.

INTRODUCTION

This poster describes several changes to be aware of before you run Release 5.18 programs under a newer release of SAS. Examples of suggested modifications are also provided.

OPTIONS AND KEYWORDS

GEN option is no longer supported. In Release 5.18, GEN specified the number of generations of history to save for each SAS data set. When converting to Release 6, delete the GEN option from the Options statement.

```
* OLD CODE *;  
OPTIONS GEN=0 NOCENTER;  
* NEW CODE *;  
OPTIONS NOCENTER;
```

TPS (terminal page size) and **TLS** (terminal line size) options are no longer supported. Replace TPS with PS (page size). Replace TLS with LS (line size).

```
* OLD CODE *;  
OPTIONS TLS=40 TPS=50;  
* NEW CODE *;  
OPTIONS LS=40 PS=50;
```

SASSUB, a method for calling a COBOL subroutine from a data step, is no longer accepted. Replace CALL SASSUB with CALL MODULE.

NON-STANDARD SYNTAX

The missing symbol “.” cannot be followed directly by a word, such as THEN, in a statement. A space is required between the missing symbol and the next word in the statement. Insert a space between “.” and the next word.

```
* OLD CODE *;  
IF X = . THEN DELETE;
```

```
* NEW CODE *;  
IF X = . THEN DELETE;
```

DO statements must have corresponding **END** statements. In Release 5.18, the END statement was not required. Add the END statement to close the DO loop at the appropriate place.

\$n.n, where n is an integer, is not recognized as a format or informat. Character informats and formats are no longer allowed to have decimal extensions. If the variable is numeric, remove the \$ sign from the format. If the variable is character, remove the decimal extension.

```
* OLD CODE *;  
FORMAT A X $6.2;  
* NEW CODE *;  
FORMAT A $6.2 X 6.2;
```

The **MISSING** option, which defines which symbol will be used to represent missing values, must not specify more than one character. Replace a string of multiple characters with a single character.

```
* OLD CODE *;  
OPTIONS MISSING = '000';  
* NEW CODE *;  
OPTIONS MISSING = '0';
```

Multiple BY statements are interpreted differently in Release 6. In Release 5.18, programs could have more than one BY statement following a PROC or SET statement. Multiple BY statements were an alternative to specifying a single BY statement with multiple variables. The following examples are equivalent under 5.18:

```
* SORT EXAMPLE 1 *;  
PROC SORT DATA=TEST;  
BY VAR1;  
BY VAR2;  
  
* SORT EXAMPLE 2 *;  
PROC SORT DATA=TEST;  
BY VAR1 VAR2;
```

In SAS Release 6, multiple BY statements are not allowed following a SET statement. If multiple BY statements follow a PROC statement, such as PROC SORT or PROC PRINT, only the last BY statement is used. Any preceding BY statements are ignored. The following examples are equivalent when run under Release 6:

```
* SORT EXAMPLE 3 *;  
PROC SORT DATA=TEST;  
BY VAR1;  
BY VAR2;  
  
* SORT EXAMPLE 4 *;  
PROC SORT DATA=TEST;  
BY VAR1;
```

Having multiple BY statements following a PROC statement does not cause a SAS error. However, multiple BY statements can produce unexpected sort results, which can lead to subsequent program errors or incorrect output. When converting to SAS Release 6, rewrite multiple BY statements as a single statement.

Division By Zero should be avoided. In Release 6.09 for MVS, attempting to divide by zero can cause a data step to process *much* more slowly than in Release 5.18.

The solution is to identify each statement in which there is division by a variable that could contain a zero value. Add a preceding statement that ensures the divisor's value will not be zero. In the following example, the variable x is set to a missing value if the divisor z is zero. This avoids the possibility of division by zero when y is divided by z.

```
* OLD CODE *;  
X = Y / Z;  
* NEW CODE *;  
IF Z = 0 THEN X = .;  
ELSE X = Y / Z;
```

DEFAULTS

FILE statements in Release 6 *overwrite* any existing contents of the specified file. Use the MOD option if you want the new output to *append* to the existing contents.

```
* OLD CODE *;  
FILE F;  
* NEW CODE *;  
FILE F(MOD);
```

PROC PRINTTO *appends* new output to any existing file contents in Release 6. Specify NEW if you want new output to *overwrite* the existing file contents instead.

```
* OLD CODE *;  
PROC PRINTTO PRINT=F;  
* NEW CODE *;  
PROC PRINTTO PRINT=F NEW;
```

Open files must be closed before they can be re-opened in a subsequent step. Use a "null" PROC PRINTTO statement to close the file before referring to it again.

```
* OLD CODE *;  
PROC PRINTTO PRINT=F;  
PROC PRINT DATA=TEST;  
DATA _NULL_;  
FILE F;  
* NEW CODE *;  
PROC PRINTTO PRINT=F;  
PROC PRINT DATA=TEST;  
PROC PRINTTO;  
DATA _NULL_;  
FILE F;
```

PROCEDURE OUTPUT

PROC PRINT output may be formatted differently in Release 6 than in Release 5.18. In output produced by

Release 6, long labels wrap to additional rows, by default, and the columns may be positioned differently.

If the output created by PROC PRINT is used as input by another application, such as a report distribution system, it might be necessary to update the page layout description in that application to correspond to the new location of columns and labels. In addition, you may notice a change in the number of observations that print on each page and in the page count.

PROC TABULATE has a change that affects report appearance. In Release 6, the BY value in the header line is surrounded by hyphens instead of blanks.

PROC FSLIST, FSEDIT and FSBROWSE default to a *bordered* screen display in Release 6. Screens designed in Release 5.18 that use all the available rows and columns will not display correctly when screen borders are added. To prevent borders from being added when converting to Release 6, use the NOBORDER option with Full Screen procedures, as shown below.

```
* OLD CODE *;  
PROC FSLIST;  
* NEW CODE *;  
PROC FSLIST NOBORDER;
```

PROC MEANS and PROC SUMMARY produce different default statistics in Release 6. In Release 5.18, nine statistics are produced if a VAR statement is not specified: n, mean, standard deviation, minimum value, maximum value, standard error of mean, sum, variance, and c.v. In Release 6, only five statistics are produced by default: n, mean, standard deviation, minimum value, and maximum value. If additional statistics are required, they can be specified in a VAR statement.

PROC CHART histograms place observations into bars differently, so the appearance Release 5.18 charts may not match the appearance of Release 6 charts. This change affects PROC CHARTs that have a VBAR or HBAR statement.

CONCLUSION

Obsolete options and keywords are usually easy to identify. If SAS Release 6 cannot interpret your Release 5.18 code, the log will usually display an error message. Other changes between Release 5.18 and Release 6 can be more subtle. They can affect how your program is processed even though no error message is displayed. To be safe, you should test programs by running the original code under Release 5.18 and the converted code under Release 6. Then carefully compare all logs, printed output, and files that are produced.

CONTACT INFORMATION

Contact the author at:

Mary MacDougall
SPS Software Services, Inc.
314 21st Street N.W.
Canton, OH 44709
Work Phone: 800-721-0809
Email: marymacd@spssoft.com