

## TS - 665

### Basics of Report Writing – What’s New?

TS-610 “Basics of Report Writing in Version 6” is an overview of writing to external files using the DATA step. The information found in TS-610 still applies in newer releases of SAS, but some new tricks and time-savers have been added as well.

This paper assumes you are familiar with the information covered in TS-610 and presents additions to DATA step report writing through Release 8.2 TSLEVEL 2M0.

#### Additions to the FILE Statement

##### DSD

The DSD option on the FILE statement enables you to write variables’ values that contain embedded delimiters using LIST output. DSD is ignored for all other types of output (i.e. formatted, column, and named). When DSD is specified, any variable’s value that contains the specified delimiter is automatically quoted with double quotation marks (") when it is written. A variable's value that does not contain the specified delimiter is not quoted. However, you can use the tilde ("~") format modifier to force any variable's value to be quoted, even if it contains no delimiter.

For example:

```
DATA ONE;
  INPUT X $ Y $ Z $;
DATALINES;
apple banana figs,red
;
```

```
DATA _NULL_;
  SET ONE;
  FILE PRINT DSD;
  PUT X ~ Y ~ Z ;
RUN;
```

```
/* RESULTS */
```

```
"apple","banana","figs,red"
```

##### DLM/ DELIMITER=

Using the DELIMITER= option, you can write data that are delimited by a specified character other than a blank. This option is valid with LIST style output only. Unlike INFILE DELIMITER= processing, only one character is used as the output delimiter.

##### FOOTNOTES

Now you can use footnotes in addition to titles. Footnotes, like titles, are global and their text can be defined outside of a DATA step. To see the footnotes in your DATA step report, make sure you specify the FOOTNOTE option on the FILE statement. The default is NOFOOTNOTES.

## ODS

Adding the ODS option to the FILE statement binds a table definition to the data component to create an output object. This object is sent to all open ODS output destinations. The fileref must be PRINT when ODS is specified on the FILE statement.

ODS is too broad a subject to be covered here. For more information see *The Complete Guide to the SAS Output Delivery System, Version 8*.

### \_FILE\_

Use the \_FILE\_ = option to assign the contents of the current output buffer to a variable. The variable is automatically retained and initialized to blanks. The maximum length of the variable is the value of the LRECL = option specified on the FILE statement.

The following example uses the variable TEMP1BUF to manipulate the current data in the output buffer.

```
DATA _null_ ;
/* routing to the output window */
FILE print _FILE_=temp1buf;
temp1buf = 'Line one in temp1';
PUT;

PUT 'TAG! You are it!' @;
fred=TEMP1buf;
SUBSTR(temp1buf,6,7) = "I'm not";
PUT;

/* routing to the log */
FILE log _FILE_=logbuf;
Logbuf="*****"||TRIM(fred)||"*****";
PUT ;
RUN;

/* RESULTS from the OUTPUT window */

Line one in temp1
TAG! I'm not it!

/* RESULTS from the LOG window */

1 DATA _null_ ;
2 FILE print _FILE_=temp1buf;
3 temp1buf = 'Line one in temp1';
4 PUT;
5 PUT 'TAG! You are it!' @;
6 fred=TEMP1buf;
7 SUBSTR(temp1buf,6,7) = "I'm not";
8
9 PUT;
10
11 FILE log _FILE_=logbuf;
12 Logbuf="*****"||TRIM(fred)||"*****";
13 PUT ;
14 RUN;
```

\*\*\*\*\* TAG! You are it! \*\*\*\*\*

NOTE: 2 lines were written to file PRINT.

NOTE: DATA statement used:

```
real time      0.49 seconds
cpu time       0.19 seconds
```

See also “New Automatic Variable” for an example using the automatic variable `_FILE_`.

### New Automatic Variable

Similar to the `_FILE_` option on the FILE statement, one can access the output buffer with the automatic variable `_FILE_`. This can be used to modify the data in the current output buffer before it is output.

Here is an example where we want to take a SAS date variable and write it out to a flat file to look like “01 January of the year 2001”.

Prior to Version 7, you can use TRIM and PUT functions to create this result:

```
DATA _NULL_;
  INPUT d mmddy6.;
  FILE PRINT;
  field=TRIM(LEFT(PUT(day(d),z2.)))||' '|| TRIM(LEFT(PUT(d,monname.)))||' of the year '|| PUT(d,year.);
  PUT field;
DATALINES;
090401
032001
;

/* RESULTS */

04 September of the year 2001
20 March of the year 2001
```

Assigning the values of `_FILE_` to a variable allows us to manipulate values (in our example, formatted versions of the values) that are in the output buffer before the buffer output, thus creating desired output with fewer function calls. Here is the example again, using `_FILE_` and an easy way to uppercase the buffer just before it is output.

```
DATA _NULL_;
  INPUT d mmddy6.;
  FILE PRINT;
  FORMAT d monname. dd z2. y year.;
  dd=day(d);
  y=d;
  PUT dd d " of the year " y @;
  field=UPCASE(_FILE_);
  PUT @1 field ;
DATALINES;
090401
032001
;

/* RESULTS */

04 SEPTEMBER OF THE YEAR 2001
20 MARCH OF THE YEAR 2001
```

### Alternative to the DATA step

If you are creating a flat file, a comma or tab delimited file for example, and you do not need a lot of control over the output, you may want to try PROC EXPORT.

The syntax is relatively straightforward, and so is the output. There are no options for titles, footnotes, headers, etc.

```
PROC EXPORT data=dsn  
  OUTFILE='c:\myfiles\report.txt'  
  DBMS=DLM;  
  DELIMITER='09'x;  
RUN;
```

The code above takes the data set WORK.DSN and creates a tab-delimited file called REPORT.TXT in the folder MYFILES, which resides on my C drive. Any single character can be specified as the delimiter. Note that the first semicolon appears after DBMS=. I put the code on 5 lines instead of 3, for ease of reading. (This is standard layout as well.)

REPORT.TXT has variable names from WORK.DSN as the first row of data. The remaining rows have the value of the variables as they were on the PDV, separated by a tab. If a SAS-provided permanent format is associated with a variable, the formatted value appears in the delimited file. (EXPORT does not handle user-defined formats. If the format is user-written, you will have to create a character variable using the PUT function and the user-defined format instead, and pass that variable to PROC EXPORT.)

To learn more about PROC EXPORT, see the SAS Procedures Guide, Version 8.

### References

SAS Language Reference: Concepts, Version 8.

SAS Language Reference: Dictionary, Version 8.