

TS-664

Janice Bloom

A DATA Stepper's Introduction to ODS

As awareness of the Output Delivery System (ODS) becomes more widespread, I am frequently asked if existing DATA _NULL_ code can simply be wrapped in ODS statements to create ODS style output. While this is a straightforward question, the answer is not.

The "Old Way"

Prior to Version 7, SAS generated output that was designed for a traditional line printer. To create a flat file or report with a DATA _NULL_ step, you could code extra information on a PUT statement related to where, how and what data was displayed. This information could include formats, line control pointers, and column control pointers. Data was "manipulated" on a *character* level.

The "New Way"

Beginning in Version 7, all output has two parts: a *template*, which defines the formatting to be used, and a *data component*, which is similar to a data set. Formatting instructions, such as variable order, text for column headings, format for columns, and stylistic references, are maintained in the template. The template is the power behind ODS. By customizing either a table or style definition of a template, one can highly customize the output.

Default Table Template

The default table template for the DATA step ('base.datastep.table') has been created and stored in the SASHELP SAS data library. Because this default table may be used for any ODS destination (Listing, HTML, printer, etc) the table is very generic. This template defines two columns - one column maps to numeric variables, and the other to character variables. The default template causes all columns to be output, in the same order as they are defined on the data object.

Key Points

1- How do you define a 'COLUMN'?

When discussing ODS, picture the *entire value* of a variable as *the column*, not the individual bytes of the value. The data component contains rows and columns of values, not rows and columns of characters.

2- If I am using a version of SAS greater than Version 6.09 on MVS or 6.12 on other platforms, I must already be using ODS to create my listing output. My list output still looks the same to me. Why are you telling me that using ODS with existing code may not produce the same results as Version 6 code?

Depending on your current code, the difference may occur when you *bind* the template to the DATA step by using ODS as an option on the FILE statement.

Show me, please!

Lets look at some DATA step examples. The examples are in pairs. The first example in each pair will not specify the ODS option on the FILE statement. The second example in the pair will be the same code, with the ODS option added to the FILE statement. I have chosen to use HTML as the output destination for the examples that bind the template to the data output.

Column pointer controls

Column pointer controls are used to place the internal pointer in specific locations on the current output line. Column control pointers in a DATA step that uses ODS moves the pointer in the same *fashion* as column pointer controls in a DATA step that does not name ODS on the FILE statement, but keep in mind in ODS a column is an entire value, not a byte. With FILE PRINT output, '+' and '@' move by characters and with FILE PRINT ODS output, they move by columns.

Character variable X and two numerics, Y and Z. DATA ONE is used in Examples 1 through 4.

```

DATA one;
  INPUT x $ y z;
DATALINES;
a 1 2
b 3 4
;

```

```
/* Example 1 */
```

DATA step code using relative column pointer control, +n.

In the example below, we write out variable X's value. With LIST style output, after the value is written out, the control pointer moves one character column to the right, by default. The +1 pointer control moves the internal pointer one more space to the right, writes out the value of Y plus one space, then moves +1 space to the right, then writes out Z. Only variable values are output. (I avoided the word "column" on purpose, to hopefully alleviate confusion.)

```

DATA _NULL_;
  SET one;
  FILE print;
  PUT X + 1 Y +1 Z;
RUN;

```

```
/* RESULTS */
```

```

a 1 2
b 3 4

```

```
/* Example 2 */
```

Use the same DATA step code as in Example 1, add ODS on the FILE statement with the default table template.

Open the HTML destination prior to the DATA step, and close it afterwards to send the output to the file named in the BODY= option. The default table definition tells SAS to include the variable names as column headers. Column controls on the PUT statement control which *column* of the data component receives a value. The PUT statement is used to output values to the data component.

By default, all columns of a data component contain the value missing. Using the PUT statement overwrites these missing values with values from the data component.

Stepping through the code along with the template, three variables or columns are defined (X, Y and Z). On the first iteration, we assign the value of X to the first table column, skip one table column, leaving it missing, move to the next table column and write the value of Y. Move one table column to the right if we can, wrap around and put out Z *. There is no more data for the rest of the line, so the final two columns remain missing.

*By default, if the ODS-column exceeds the number of columns in the data component, ODS writes the current line, moves the pointer to the first column on the next line and continues to process the PUT statement. See page 113 and 107 in The Complete Guide to the SAS Output Delivery System, Version 8 for more details and options to change this behavior.

```
ODS HTML body="c:\tracks\ods\examples\relcontrol.html";
```

```

DATA _NULL_;
  SET one;
  FILE print ODS;
  PUT X + 1 Y +1 Z;
RUN;

```

```
ODS HTML close;
```

```
/* HTML Table RELCONTROL */
```

x	y	z
a	.	1
2	.	.
b	.	3
4	.	.

```
/* Example 3 */
```

This example is of the commonly used absolute control pointer, @n. The value of X is written to text-column one, and the value of Y is written to text-column 5. Only the values of X and Y will be written to the output window.

```
DATA _null_;
  SET one;
  FILE PRINT;
  PUT @1 x @5 y;
RUN;
```

```
/* RESULTS */
```

```
a 1
b 3
```

```
/* Example 4 */
```

Same DATA step using absolute column pointer controls, with ODS on the FILE statement and the default template. This time we are moving by "table" columns. @1 moves to the first defined column and @5 tries to move to the fifth column, if one is defined. In this example there are only two variables, X and Y, not enough for five, so we move to the first available column, which is on a new line.

```
ODS HTML body="C:\tracks\ods\examples\abscolpointers.htm";
```

```
DATA _NULL_;
  SET one;
  FILE PRINT ODS;
  PUT @1 x @5 y;
RUN;
```

```
ODS HTML close;
```

```
/* HTML Table ABSCOLPOINTERS */
```

x	y	z
a	.	.
1	.	.
b	.	.
3	.	.

Line Control Pointers

Line pointer controls specify the output line where the PUT statement is to write a value. As with column pointer controls, the syntax is the same for line pointer controls whether they are used in a DATA step that specifies ODS on the FILE statement, or in a DATA step that does not.

```
/* DATA num contains three numeric variables and is used for Examples 5 and 6 */
```

```
DATA num;  
  INPUT x y z;  
DATALINES;  
1 2 3  
4 5 6  
;
```

```
/* Example 5 */
```

With this DATA step code, the output will start in column (byte) one. Variable values for X, Y and Z will be output and the slash pointer control will force SAS to skip one blank line after Z is written out.

```
DATA _NULL_;  
  SET num;  
  FILE PRINT notitles;  
  PUT x y z / ;  
RUN;
```

```
/* Results */
```

```
1 2 3  
  
4 5 6
```

```
/* Example 6 */
```

Remember that with ODS, all of the columns in the data component contain a missing value, and the PUT statement is used to *overwrite* these missing values. By skipping a line, the missing values do not get overwritten, and will appear in the resulting output*. Again, the default table template calls for the variable names to be output. The variable names are centered, and the values line up to the right of the column. Using a null PUT instead of a slash produces the same results.

(* You can use the MISSING= option to specify missing numerics be represented by a space instead of a dot, so final output appears more like Example 5.)

```
ODS HTML body="C:\tracks\ods\examples\linepointers.htm";
```

```
DATA _NULL_;  
  SET num;  
  FILE PRINT ODS notitles;  
  PUT x y z / ;  
RUN;
```

```
ODS HTML close;
```

```
/* HTML table LINEPOINTERS */
```

x	y	z
1	2	3
.	.	.
4	5	6
.	.	.

Another line pointer control, PUT _PAGE_, is used to force SAS to a new page upon execution. If PUT _PAGE_ is used in ODS, the style used needs to "recognize" the concept of pages before code can force a new page.

This paper has been comparing non-ODS output to HTML style ODS output. When we discuss pagebreaks, the comparison gets a little "unfair". HTML is not really designed to have pagebreaks. HTML is viewed through your web browser, on your computer screen. Generally, you scroll up and down or right to left to see the entire page. You can print from a web page, but have probably experienced printed output that is not always what you saw on the computer screen. If your target audience will be printing your output, you may be more successful if you use a "paper-based" file format like Adobe Acrobat's PDF format (PDF style is supported in Version 8.2) or postscript (PS or PCL, supported in Version 7/8).

(ODS FAQ, http://www.sas.com/rnd/base/topics/templateFAQ/Template_csstyle.html#pb)

Data PRODUCE has one variable called FRUIT. FRUIT has three BY groups: apple, banana, and coconut. This data set will be used for Examples 7 and 8.

```
DATA produce;
  INPUT fruit $ ;
DATALINES;
apple
apple
banana
coconut
coconut
coconut
;
```

```
/* Example 7 */
```

This example uses first.logic to force a new page for each new value of FRUIT. The PAGESIZE value is set to 20, and FORMDLIM is "*" for demonstration purposes.

```
OPTION PS=20 FORMDLIM="*";
```

```
DATA _NULL_;
  FILE print notitle;
  SET produce;
  BY fruit;
  IF first.fruit then put _page_;
  PUT fruit;
RUN;
```

```

/* Results */
*****

apple
apple

*****

banana

*****

coconut
coconut
coconut

/* Example 8 */

Same DATA step code plus ODS specified on the FILE statement.

ODS HTML body="C:\tracks\ODS\examples\forcepage.htm";

DATA _NULL_;
  FILE print notitle ODS;
  SET produce;
  BY fruit;
  IF first.fruit then put _page_;
  PUT fruit;
RUN;

ODS HTML close;

/* HTML table FORCEPAGE is too large to be included here */

```

Recommended Reading:

Haworth, Lauren E. Output Delivery System: The Basics, Cary, NC: SAS Institute Inc, 2001.

Heffner, William F. (1998), "ODS: The Data Step Knows," in the Proceedings for the Twenty-Third Annual SAS Users Group International Conference, Cary, NC: SAS Institute Inc.

Olinger, C.(2000), "ODS for Dummies," in the Proceedings for the Twenty-Fifth Annual SAS Users Group International Conference, Cary, NC: SAS Institute Inc.

The Complete Guide to the SAS Output Delivery System, Version 8, Cary, NC: SAS Institute Inc, 1999.

Web links:

Research and Development -- <http://support.sas.com/rnd/base/ods/index.html>
 PROC TEMPLATE FAQ -- <http://support.sas.com/rnd/base/ods/templateFAQ/index.html>

Reference:

Heffner, William F. (1998), "ODS: The Data Step Knows," in the Proceedings for the Twenty-Third Annual SAS Users Group International Conference, Cary, NC: SAS Institute Inc.

The Complete Guide to the SAS Output Delivery System, Version 8, Cary, NC: SAS Institute Inc, 1999.

ODS FAQ, http://www.sas.com/rnd/base/topics/templateFAQ/Template_csstyle.html#pb