Macros and Conventional Macro Variables: Effective Tools for Customizing Tabular Output in SAS® ODS

Perry Watts, Independent Consultant, Elkins Park, PA

ABSTRACT

While it is understood that macro variables are referenced in ODS by using MVAR and NMVAR statements, macro programs can also play a vital role in ODS. To show how they work, the TABLETEMPLATE macro that generates a customized table in ODS is reviewed in depth. Looping in the macro eliminates the need for hard coding, and multiple ampersand macro variables **(&&f&i)** are used to insert data dependent text strings into a table. Pre and Post macro versions of the code are presented to emphasize the role played by macro programs in ODS.

PROBLEM DEFINITION

To generalize PROC TEMPLATE table definitions, it is recommended that DYNAMIC variables reference structural components such as headers or footers and that MVAR and NMVAR be reserved for constants that can change each time a compiled template is called in a data step. However, what if the macro variables you need to define come from the set of unique values in a column from the input data? Furthermore, what if you need a different outcome in a PROC TEMPLATE CELLSTYLE-AS statement for each unique value in the column? In this situation, a reliable table definition requires the assistance of a full-fledged macro with looping capabilities. What the paper describes is an application where an ODS table definition is nested inside the macro TABLETEMPLATE. The table definition uses MVAR and DYNAMIC variables in addition to conventional macro variables with multiple ampersands for managing the CELLSTYLE-AS statement.

TRACING FONTS IN THE ODS STYLES.DEFAULT TEMPLATE FOR VERSION 9.1.3 SAS

Colors and fonts represented over 50% of the 184 attributes that are associated with style elements that have CONTAINER as a common ancestor in the ODS Styles.Default template. Knowing which color or font is actually used in a given style element can be difficult to determine because of a convoluted mapping process that is embedded in the template. The mapping process is illustrated in Figure 1 below.

Figure 1. A partial listing of a numbered version of the ODS Styles.Default template is presented in linear order. Fonts are referenced by name rather than by value throughout the template. This means you have to go back to the beginning of the template to find out that the default value for DOCFONT in the CONTAINER style element is Arial, relative size 3. Helvetica is only used if Arial is not available, and a sans-serif font is used if both Arial and Helvetica are not available. (Gray-overlay fonts below are not used by any of the CONTAINER style elements).

```
Obs                                  Style Definition
---    ------------------------------------------------------------------------------
  1       style fonts
  2         "Fonts used in the default style" /
  3         'TitleFont2' = ("Arial, Helvetica, sans-serif",4,Bold Italic)
  4         'TitleFont' = ("Arial, Helvetica, sans-serif",5,Bold Italic)
  5         'StrongFont' = ("Arial, Helvetica, sans-serif",4,Bold)
  6         'EmphasisFont' = ("Arial, Helvetica, sans-serif",3,Italic)
  7         'FixedEmphasisFont' = ("Courier New, Courier, monospace",2,Italic)
  8         'FixedStrongFont' = ("Courier New, Courier, monospace",2,Bold)
  9         'FixedHeadingFont' = ("Courier New, Courier, monospace",2)
 10         'BatchFixedFont' = ("SAS Monospace, Courier New, Courier, monospace",2)
 11         'FixedFont' = ("Courier",2)
 12         'headingEmphasisFont' = ("Arial, Helvetica, sans-serif",4,Bold Italic)
 13         'headingFont' = ("Arial, Helvetica, sans-serif",4,Bold)
 14         'docFont' = ("Arial, Helvetica, sans-serif",3);
...
164       style Container
165         "Abstract. Controls all container oriented elements." /
166         font = Fonts('DocFont')
167         foreground = colors('docfg')
168         background = colors('docbg');
...
367       style Batch from Output
368         "Controls batch mode output." /
369         font = fonts('BatchFixedFont')
370         foreground = colors('batchfg')
371         background = colors('batchbg');
```

What is needed in this situation is a table that links a font directly to the style element that uses it. In Figure 2, a few records from the input data set, FONTTRACER, derived from the styles.default template, are matched to the corresponding table entries that are produced from an application of the TABLETEMPLATE macro. What gives the table its pizzazz is that the font attribute is formatted by the referenced font. There is a one-to-one correspondence between the FONT column in the font tracer and the SAMPLE column in the color tracer. Both tables are included in the appendix of this paper. Complete source code for the font tracer is also included as an attachment to the paper in the proceedings.

Figure 2. The input data set is transferred into a font tracer table with an application of the TABLETEMPLATE macro. By looking at a display of the actual fonts in the table, a more informed decision can be made when attributes need to be changed in a style element.

Input Data Set: fontTracer

```
Obs  font                                                fontName         styleName

 1   "sas monospace, courier new, courier, monospace",2   batchfixedfont   Batch
 2   "arial, helvetica, sans-serif",3                     docfont          Container
...
38   "arial, helvetica, sans-serif",5,bold italic         titlefont        SystemTitle
39   "arial, helvetica, sans-serif",5,bold italic         titlefont        SystemFooter
40   "arial, helvetica, sans-serif",4,bold italic         titlefont2       TitlesAndFooters
```

Font	Font Name	Style Element
"sas monospace, courier new, courier, monospace",2	batchfixedfont	Batch
"arial, helvetica, sans-serif",3	docfont	Container
"arial, helvetica, sans-serif",5,bold italic	titlefont	SystemTitle
		SystemFooter
"arial, helvetica, sans-serif",4,bold italic	titlefont2	TitlesAndFooters

THE PRE-MACRO VERSION OF THE SOURCE CODE FOR THE FONT TRACER

Yellow highlighted code in the listing below is changed in the TABLETEMPLATE macro. The repetitive code that is highlighted indicates the need for a macro program. Gray highlights reference MVAR and DYNAMIC variables. To see how COLHD is actually used, look at the DATA _NULL_ step in the full version of the source code included in the proceedings. The DYNAMIC statement is also covered on pages, 342,348, and 391 in the ODS User's Guide **[2].**

```
/* CREATE MACRO VARIABLES */
proc sql noprint;
  create table uniqueFonts as
    select distinct font from fontTracer;
  select count(*) into :n
    from uniqueFonts;
  select font into :f1 - :f%left(&n)
    from uniqueFonts;
quit;
%let n = %left(%trim(&n));

/* CREATE A TABLE TEMPLATE */
proc template;
  define Table TABLES.fontTbl;
    dynamic colhd;
    mvar f1 f2 f3 f4 f5 f6 f7 f8 f9;
    classlevels=on;
    define column columnFont;
      blank_dups=on;
      define header colhd;
        style=header;
        just=Center;
      end;
      header=colhd;
      CELLSTYLE
        _val_ EQ F1 as {font = ("arial, helvetica, sans-serif",3)},
        _val_ EQ F2 as {font = ("arial, helvetica, sans-serif",3,italic)},
        _val_ EQ f3 as {font = ("arial, helvetica, sans-serif",4,bold)},
        _val_ EQ f4 as {font = ("arial, helvetica, sans-serif",4,bold italic)},
```

```
            _val_ EQ f5 as {font = ("arial, helvetica, sans-serif",5,bold italic)},
            _val_ EQ f6 as {font = ("courier new, courier, monospace",2,bold)},
            _val_ EQ f7 as {font = ("courier new, courier, monospace",2,italic)},
            _val_ EQ f8 as {font = ("courier",2)},
            _val_ EQ F9 as {font = ("sas monospace, courier new, courier, monospace",2)},
            1 AS data;
        end;  /* DEFINE COLUMN */
        define column restOfData;
          generic=on;
          blank_dups=on;
          define header colhd;
            style=header;
            just=Center;
          end;
          header=colhd;
        end; /* DEFINE COLUMN */
      end;   /* DEFINE TABLE */
    run;
```

The CELLSTYLE-AS statement is very similar to the SELECT statement in base SAS. Each time a record is processed, _VAL_ is compared to the contents of F1-F9. When a match is found, ODS applies the corresponding font to the output and moves on to the next record. If there is no match, the default font associated with the DATA style element is applied to the output. (**`1 AS data`** in the code plays the same role as OTHERWISE in a SELECT statement).

As a preliminary step, it is necessary to print out the data set UNIQUEFONTS in order to hard-code the values for the MVAR variables. Hard-coding is required for conformance to style-attribute specification in the syntax:

*CELLSTYLE expression AS {style-attribute-**name** = style-attribute-**value**}* **[2, p. 389].**

The *style-attribute-**value*** cannot reference a variable, not even an MVAR variable. It only works with text strings, and in this program, the text string on the right side of the equation is equivalent to the MVAR variable in the expression. Thus:

```
_val_ EQ F1 as {font = ("arial, helvetica, sans-serif",3)}
```

translates to:

```
_val_ EQ "arial, helvetica, sans-serif",3 as {font = ("arial, helvetica, sans-serif",3)}
```

This is how the font attribute is formatted by the referenced font in Figure 2.

CELLSTYLE-AS is typically used to check data set values against ranges, not point values. *Output Delivery System: The Basics and Beyond*, shows by example how to assign traffic lighting by specifying ranges with user-defined endpoints in a CELLSTYLE-AS statement **[1, p. 282]**.

USING THE TABLETEMPLATE MACRO TO GENERATE THE FONT TRACER

Below is the revised code that replaces hard-coding with the TABLETEMPLATE macro. Not only is the code shorter in length, it is also more reliable. There is no need, for example, to obtain intermediate printouts for the number and values of the MVAR variables. Also, there is no need for the catchall **`1 as data`** in the CELLSTYLE-AS statement. The data set FONTTRACER supplies values to the macro variables from the same column that is being processed in PROC TEMPLATE.

```
/* CREATE MACRO VARIABLES */
proc sql noprint;
  create table uniqueFonts as
    select distinct font from fontTracer;
  select count(*) into :n
    from uniqueFonts;
  select font into :f1 - :f%left(&n)
    from uniqueFonts;
quit;
%let n = %left(%trim(&n));

/* CREATE A TABLE TEMPLATE */
%macro tableTemplate;
  proc template;
    define Table TABLES.fontTbl;
      dynamic colhd;
      mvar %do i = 1 %to &n; f&i %end; ;
      classlevels=on;
      define column columnFont;
        blank_dups=on;
        define header colhd;
          style=header;
          just=Center;
        end;
```

```
        header=colhd;
        cellstyle
          %do i= 1 %to %eval(&n - 1);
            _val_ EQ f&i as {font = (&&f&i)},
          %end;
          _val_ EQ f&n as {font = (&&f&n)};
      end;  /* DEFINE COLUMN */
      define column restOfData;
        generic=on;
        blank_dups=on;
        define header colhd;
          style=header;
          just=Center;
        end;
        header=colhd;
      end; /* DEFINE COLUMN */
    end;   /* DEFINE TABLE */
  run;
%mend tableTemplate;
%tableTemplate
```

SUMMARY AND CONCLUSIONS

The TABLETEMPLATE macro described in this paper shows that macro programs can play a vital role in the construction of ODS table templates. With this macro, repetitive coding is eliminated and program reliability is increased. The macro also makes it possible to extend the capability of the CELLSTYLE-AS statement to check for point value equalities. In addition, greater transparency is achieved with TABLETEMPLATE. The relationship between an MVAR variable and font assignment is clarified with `_val_ EQ F&i as {font = (&&F&i)}` in the CELLSTYLE-AS statement.

COPYRIGHT STATEMENT

REFERENCES

[1] Haworth, Lauren E., Cynthia L. Zender, and Michele M. Burlew. *Output Delivery System: The Basics and Beyond.* Cary, NC: SAS Institute Inc., 2009.

[2] SAS Institute Inc. *SAS® 9.1 Output Delivery System: User's Guide.* Cary NC: SAS Institute Inc., 2004.

TRADEMARK CITATION

CONTACT INFORMATION

The author welcomes feedback via email at mailto:perryWatts@comcast.net

APPENDIX

1) FONT TRACER

Font Tracer for the SAS STYLES.DEFAULT Container Style Elements

Font	Font Name	Style Element
"sas monospace, courier new, courier, monospace",2	batchfixedfont	Batch
"arial, helvetica, sans-serif",3	docfont	Container
"arial, helvetica, sans-serif",3,italic	emphasisfont	IndexTitle
		ExtendedPage
		DataEmphasis
		HeaderEmphasis
		RowHeaderEmphasis
		FooterEmphasis
		RowFooterEmphasis
"courier new, courier, monospace",2,italic	fixedemphasisfont	DataEmphasisFixed
		HeaderEmphasisFixed
		RowHeaderEmphasisFixed
		FooterEmphasisFixed
		RowFooterEmphasisFixed
"courier",2	fixedfont	NoteContentFixed
		WarnContentFixed
		ErrorContentFixed
		FatalContentFixed
		DataFixed
		HeaderFixed
		RowHeaderFixed
		FooterFixed
		RowFooterFixed
"courier new, courier, monospace",2,bold	fixedstrongfont	ProcTitleFixed
		DataStrongFixed
		HeaderStrongFixed
		RowHeaderStrongFixed
		FooterStrongFixed
		RowFooterStrongFixed
"arial, helvetica, sans-serif",4,bold	headingfont	Byline
		HeadersAndFooters
	strongfont	PageNo
		DataStrong
		HeaderStrong

Font	Font Name	Style Element
		RowHeaderStrong
		FooterStrong
		RowFooterStrong
"arial, helvetica, sans-serif",5,bold italic	titlefont	SystemTitle
		SystemFooter
"arial, helvetica, sans-serif",4,bold italic	titlefont2	TitlesAndFooters

2) COLOR TRACER

Color Tracer for the SAS STYLES.DEFAULT Container Lineages

COLORS: (FG=Foreground BG=Background)			Style Elements that Use the Colors			
Color	Abbreviation	Location	#1	#2	#3	#4
cxd3d3d3	bga3	batchbg	Batch			
cx000000	fga1	batchfg	Batch			
cxb0b0b0	bga2	bylinebg	Byline			
cx0033aa	fga2	bylinefg	Byline			
cxe0e0e0	bga	captionbg	Caption			
cx000000	fga1	captionfg	Caption			
cx0033aa	fga2	conentryfg	IndexItem			
cx002288	fga	confolderfg	ContentFolder			
cxb0b0b0	bga2	contentbg	Index	Contents	Pages	Date
cx0033aa	fga2	contentfg	Index	Contents	Pages	Date
cx002288	fga	contitlefg	IndexProcName	IndexTitle		
cxd3d3d3	bga3	databg	Data			
cxd3d3d3	bga3	databgemph	DataEmphasis			
cxd3d3d3	bga3	databgstrong	DataStrong			
cx000000	fga1	datafg	Data			
cx000000	fga1	datafgemph	DataEmphasis			
cx000000	fga1	datafgstrong	DataStrong			
cxe0e0e0	bga	docbg	Container	BodyDate	BylineContainer	
cx002288	fga	docfg	Container	BodyDate		
cxb0b0b0	bga2	headerbg	HeadersAndFooters			

COLORS: (FG=Foreground BG=Background)			Style Elements that Use the Colors			
Color	Abbreviation	Location	#1	#2	#3	#4
cxb0b0b0	bga2	headerbgemph	HeaderEmphasis			
cxb0b0b0	bga2	headerbgstrong	HeaderStrong			
cx0033aa	fga2	headerfg	HeadersAndFooters			
cx0033aa	fga2	headerfgemph	HeaderEmphasis			
cx0033aa	fga2	headerfgstrong	HeaderStrong			
cx004488	fgb1	link1	Document			
cx0066aa	fgb2	link2	Document			
cxe0e0e0	bga	notebg	Note			
cx002288	fga	notefg	Note			
cxe0e0e0	bga	proctitlebg	ProcTitle			
cx002288	fga	proctitlefg	ProcTitle			
cxe0e0e0	bga	systitlebg	TitlesAndFooters			
cx002288	fga	systitlefg	TitlesAndFooters			
cxf0f0f0	bga1	tablebg	Output			
cx000000	fga1	tableborder	Output			