Paper DV07-2012

ODS ExcelXP: Customizing Cell Pattern, Borders and Cell Indenting

Deepak Asrani, Medtronic Inc., Mounds View, MN, USA

ABSTRACT

ODS Excelxp tagset offers the most flexible way to write SAS output to Microsoft Excel. Standard customizable options in ExcelXP tagset handle most of the excel formatting very well. However, given the ever growing formatting features in Excel, there is always a need to go beyond what is available. This paper describes 'how-to' of some of the Excel formatting that requires modification of standard ExcelXP tagset. Specifically, this paper concentrates on customizing cell pattern, borders and cell indenting.

INTRODUCTION

An ExcelXP tagset consists of events that write various sections of the final Excel document in XML format. The tagset parses the TAGATTR values and applies them through the various events. There are three different methods for customization of cell formats

- The value will directly affect the attributes for the cell e.g. Background color of the cell
- The value may go into a new style definition that the cell will then reference e.g. Customization the Cell pattern
- The value of TAGATTR= changes the XML for the row but not the cell e.g. Hidden

The approach used by the author in identifying and customizing the cell formats includes following steps

1. Creating an excel sheet with the cell formatting that needs to be customized and saving the sheet in XML format.
2. Identifying the section of XML that causes the cell formatting to take place.
3. Identifying the event within ExcelXP tagset that writes the XML section.
4. Creating a custom ExcelXP tagset that modifies that event.
5. Testing. Testing.Testing.

CUSTOMIZING CELL PATTERN

Although ExcelXP tagset offers multiple ways to format foreground and background of cell contents, a user may still find need for additional formatting. A typical situation in life science world is when a cell falls into 2 or more different categories. For e.g., A blood pressure measurement may be the first measurement and also may be the single largest amongst a series of measurements. The cell containing the value may be needed to have a background color of yellow to indicate first measurement and have gray dot patterns to indicate the largest measurement.

A new tagattr PATTERN needs to be defined.

```
DEFINE EVENT compile_regexp;

...

...

set $tagattr_regexp "/^([Ff][Oo][Rr][Mm][Aa][Tt]:|[Ff][Oo][Rr][Mm][Uu][Ll][Aa]:|";
set $tagattr_regexp $tagattr_regexp "[Rr][Oo][Tt][Aa][Tt][Ee]:|[Tt][Yy][Pp][Ee]:|";
set $tagattr_regexp $tagattr_regexp "[Hh][Ii][Dd][Dd][Ee][Nn]:|";
set $tagattr_regexp $tagattr_regexp "[Pp][Aa][Tt][Tt][Ee][Rr][Nn]:|";
set $tagattr_regexp $tagattr_regexp "[Mm][Ee][Rr][Gg][Ee][Aa][Cc][Rr][Oo][Ss][Ss]:)/";
eval $tagattr_regex prxparse($tagattr_regexp);

 END;
```

Modify the event FONT_INTERIOR which writes the XML code for the cell interior to use the PATTERN passed on by the user.

```
DEFINE EVENT font_interior;

...

...

  do /if cmp( htmlclass, "pagebreak");
```

```
      stop /if ^any( background, tagattr, foreground);
      put "<Interior";
      putq " ss:Color=" BACKGROUND;
      putq " ss:Pattern=" tagattr;
      putq " ss:PatternColor=" FOREGROUND /if (tagattr);
      put " />" NL;

   else;

      do /if background;
         put "<Interior";

         do /if ^cmp( background, "transparent");
            putq " ss:Color=" BACKGROUND;
           do /if $attrs["pattern"];
               set $pattern_override propcase($attrs["pattern"]);
              putq " ss:Pattern=" strip($pattern_override)  /if exist( BACKGROUND);
            else;
               put " ss:Pattern=""Solid""" /if exist( BACKGROUND);
           done;
         done;

         put " />" NL;

       else /if cmp( htmlclass, "body");
         put "<Interior ss:Pattern=""Solid"" />" NL;
       done;
  done;

END;
```

Define a style based on the new tag attribute.

```
style yellowdots from data/background=yellow;
style yellowdots from yellowback/ tagattr="Pattern:Gray125";
```

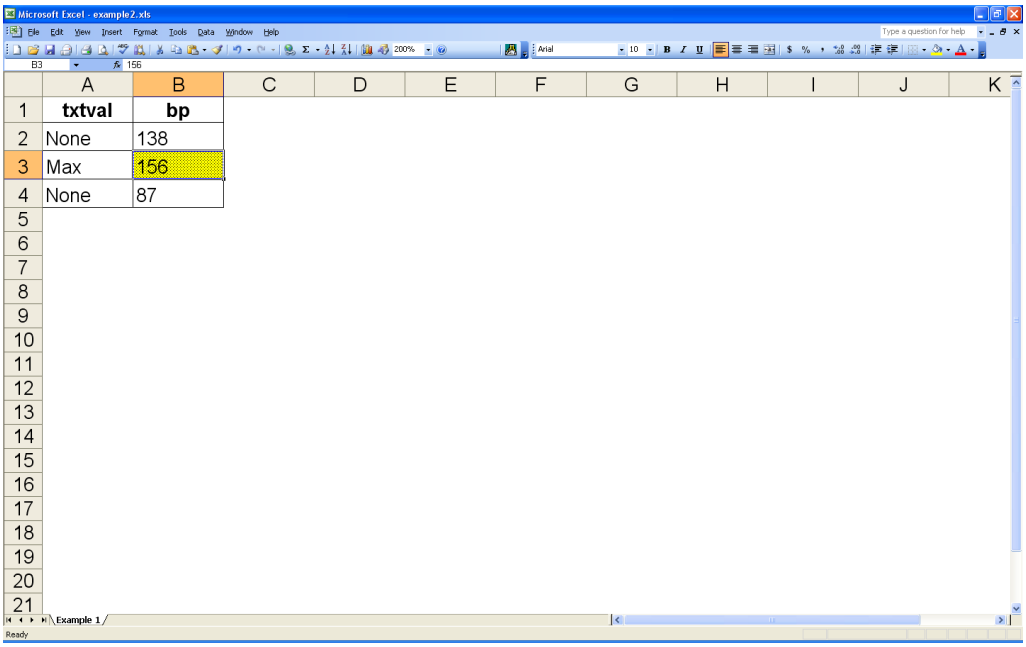
Now consider the dataset below:

txtval	bp
None	138
First and Max	156
None	87

Table 1. Table used for Example1 and Example 2

If the following code is executed on the above table, the bp value of 156 will have a background color of yellow and a gray pattern with 12.5% gray.

```
ODS TAGSETS.CUSTOMEXCELXP STYLE=styles.slsrep10 options(sheet_name="Example 1");
PROC REPORT DATA=example1 NOWD;
    COLUMN txtval bp;
    DEFINE txtval / DISPLAY;
    DEFINE bp / DISPLAY;
    COMPUTE bp;
          IF bp=156 then CALL DEFINE("bp","style","STYLE=yellowdots");
    ENDCOMP;
RUN;
ODS tagsets.customexcelcp CLOSE;
```



Display 1. Excel Output with Customized Cell Pattern

CUSTOMIZING CELL BORDERS

The ExcelXP formatting for cell borders is limited in the sense that only one set of borders can be applied to all four borders of the cell. The cell borders default to CONTINUOUS if nothing is specified. The event that affects the border style is GET_BORDERSTYLE

```
DEFINE EVENT get_borderstyle;
       set $borderstyle borderrightstyle / if cmp( $border_position, "Left");
       set $borderstyle bordertopstyle / if cmp( $border_position, "Right");
       set $borderstyle borderbottomstyle / if cmp( $border_position, "Top");
       set $borderstyle borderleftstyle / if cmp( $border_position, "Bottom");
        set $borderstyle upcase($borderstyle);
        set $borderstyle $borderstyles[$borderstyle];

       do /if ^$borderstyle;
          set $borderstyle "Continuous" /if $borderwidth;
        done;
END;
```

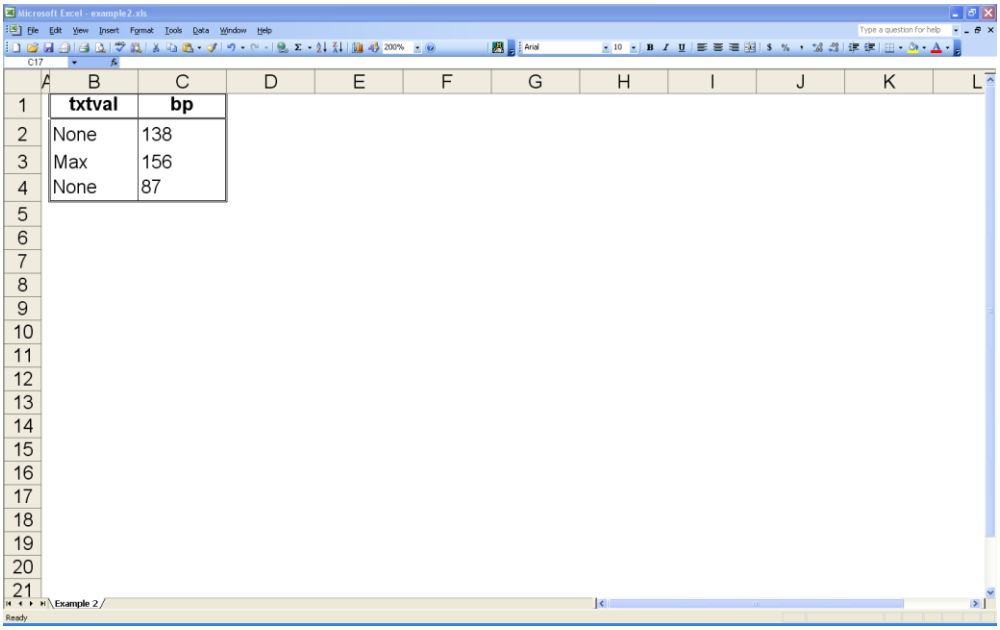
Next define the styles that pass on the required borders to the tagset. The following code puts double border around the table and header, and a single border in between other columns.

```
       style header_top_bottom from header /
               borderleftstyle=solid
                borderrightstyle=solid
                bordertopstyle=double
                borderbottomstyle=double;

       style header_right from header /
                borderleftstyle=double
                borderrightstyle=solid
                bordertopstyle=double
                borderbottomstyle=double;

       style header_left from header /
                borderleftstyle=solid
                borderrightstyle=double
```

```
                bordertopstyle=double
                borderbottomstyle=double;
```

Using the dataset from Table1 and on execution of following code, the output will have header and table with double lines and column separators that are single.

```
ODS TAGSETS.CUSTOMEXCELXP STYLE=styles.slsrep10 options(sheet_name="Example 2");
PROC REPORT DATA=example1 NOWD;
COLUMN txtval bp;
DEFINE txtval/DISPLAY STYLE(HEADER)=header_right STYLE(COLUMN)=data_left;
DEFINE bp / DISPLAY STYLE(HEADER)=header_left STYLE(COLUMN)=data_right;
COMPUTE bp;
IF bp=87 THEN DO;
   CALL DEFINE(_ROW_, "STYLE","STYLE={borderleftwidth=1px borderrightwidth=1px
   borderbottomstyle=double borderbottomwidth=1px}");
   CALL DEFINE(_COL_,"STYLE","STYLE={borderrightstyle=double borderrightwidth=1px}");
END;
ENDCOMP;

COMPUTE txtval;
   CALL DEFINE(_COL_,"STYLE","STYLE={borderleftstyle=double borderleftwidth=1px}");
ENDCOMP;
RUN;
ODS TAGSETS.CUSTOMEXCELXP CLOSE;
```



Display 2. Excel Output with Customized Cell Borders

CUSTOMIZING CELL INDENTING

INSERT SPACE BEFORE CELL CONTENTS

ExcelXP tagset takes care of certain characters in the standard tagset code. These special characters include those characters that are used to define the body of xml like <, > and &. Indenting of cells can be achieve by adding another special character ~ (tilde) to the list which is resolved to single space in the XML output produced. Note that the ASCII code for space is 32.

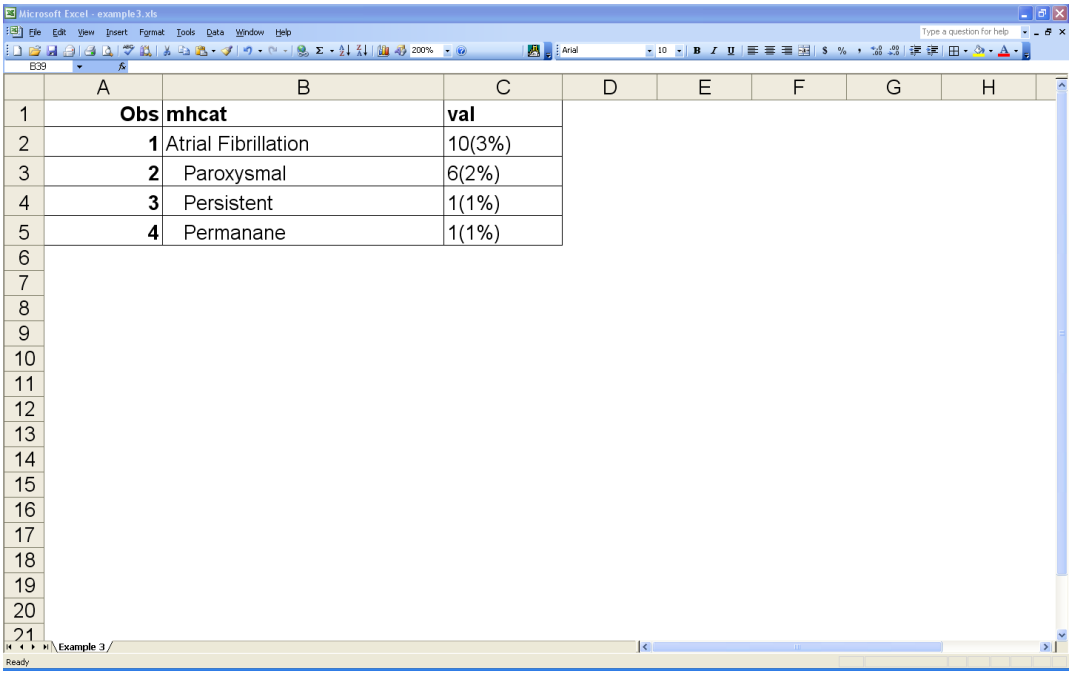
```
map = %NRSTR("<>&~");
mapsub = %NRSTR("/&lt;/&gt;/&amp;/&#32;/");
```

Now apply the above code to a dataset below:

mhcat	val
Atrial Fibrillation	10(3%)
~~Paroxysmal	6(2%)
~~Persistent	1(1%)
~~Permanane	1(1%)

Table 2. Table used for Example3

The result is



	A	B	C	D	E	F	G	H
1	**Obs**	**mhcat**	**val**					
2	**1**	Atrial Fibrillation	10(3%)					
3	**2**	Paroxysmal	6(2%)					
4	**3**	Persistent	1(1%)					
5	**4**	Permanane	1(1%)					

Display 3. Excel Output with Customized Cell Indenting

INSERT A 'SOFT RETURN' IN CELL

Breaking the cell contents into multiple rows within the same cell can be achieved by inserting a 'soft return' in the cell contents. The idea is similar to the one used for cell indenting. Add a special character | (pipe) which is resolved to soft return in the XML output. Note that the ASCII code for soft return is 10.

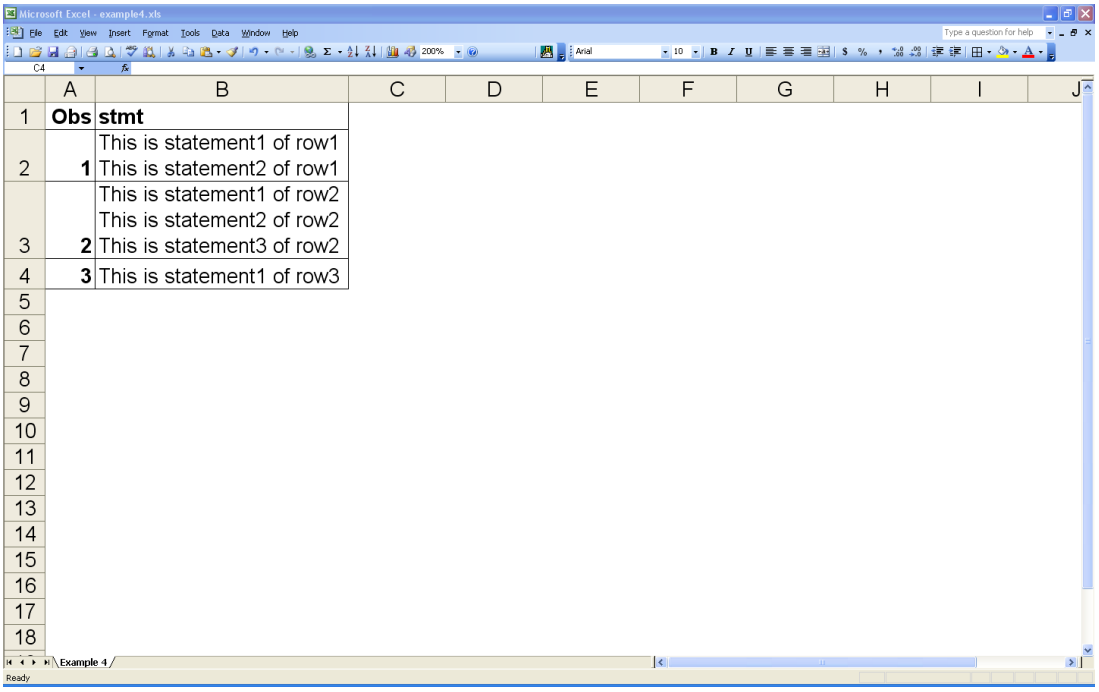
```
map = %NRSTR("<>&|");
mapsub = %NRSTR("/&lt;/&gt;/&amp;/&#10;/");
```

Now apply the above code to a dataset below:

stmt
This is statement1 of row1|This is statement2 of row1
This is statement1 of row2|This is statement2 of row2|This is statement3 of row2
This is statement1 of row3

Table 3. Table used for Example4

The result is



	A	B	C	D	E	F	G	H	I	J
1	**Obs**	**stmt**								
2	**1**	This is statement1 of row1 This is statement2 of row1								
3	**2**	This is statement1 of row2 This is statement2 of row2 This is statement3 of row2								
4	**3**	This is statement1 of row3								
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										

Display 4. Excel Output with Multiple rows per cell

CONCLUSION

In this paper we have explored a drop in the ocean of what ExcelXP tagset has to offer. The capabilities of user defined TAGATTR are endless and this is what makes the ExcelXP really flexible. The approach of looking at XML and investigating the section that generates that XML in ExcelXP tagset is all that is needed to make the ExcelXP tagset truly extensible. Thank you SAS® for putting this excellent tool in the hands of developers!!!

APPENDIX

```
DATA example1;
LENGTH txtval $100;
INPUT bp $ txtval $ &;
DATALINES;
138 None
156 First and Max
87 None
;

DATA example3;
LENGTH mhcat $20 val $7;
INPUT mhcat $1-20 val $21-26;
DATALINES;
Atrial Fibrillation 10(3%)
~~~Paroxysmal        6(2%)
~~~Persistent        1(1%)
~~~Permanane         1(1%)
;

DATA example4;
LENGTH stmt $100;
INPUT stmt $ &;
DATALINES;
This is statement1 of row1|This is statement2 of row1
This is statement1 of row2|This is statement2 of row2|This is statement3 of row2
This is statement1 of row3
;
```

```
%macro CreateCustomExcelXP;
       ods path work.templat(update)
              sasuser.templat(update)
              sashelp.tmplmst(read);


       PROC TEMPLATE;
         DEFINE STYLE styles.slsrep10;
       parent = styles.default;
       style header/
           font_weight = bold
                font_face=Arial
              font_size=10pt
           foreground = black
                background=white
                borderstyle=hidden
                borderleftstyle=hidden
                borderrightstyle=hidden
                bordertopstyle=hidden
                borderbottomstyle=hidden
                borderleftcolor=black
                borderrightcolor=black
                bordertopcolor=black
                borderbottomcolor=black
                borderleftwidth=1px
                borderrightwidth=1px
                bordertopwidth=1px
                borderbottomwidth=1px;

       style header_top_bottom from header /
                borderleftstyle=solid
                borderrightstyle=solid
                bordertopstyle=double
                borderbottomstyle=double;

       style header_right from header /
                borderleftstyle=double
                borderrightstyle=solid
                bordertopstyle=double
                borderbottomstyle=double;

       style header_left from header /
                borderleftstyle=solid
                borderrightstyle=double
                bordertopstyle=double
                borderbottomstyle=double;

       style data from data /
                font_face=Arial
              font_size=10pt
                foreground = black
                background=white
                borderstyle=hidden
                borderleftstyle=hidden
                borderrightstyle=hidden
                bordertopstyle=hidden
                borderbottomstyle=hidden
                borderleftcolor=black
                borderrightcolor=black
                bordertopcolor=black
                borderbottomcolor=black
                borderleftwidth=1px
                borderrightwidth=1px
```

```
                bordertopwidth=0px
                borderbottomwidth=0px;

        style data_left from data /
                borderleftstyle=double
                bordertopstyle=hidden
                borderbottomstyle=hidden
                borderrightstyle=solid;

        style data_right from data /
                borderleftstyle=solid
                bordertopstyle=hidden
                borderbottomstyle=hidden
                borderrightstyle=double;

        style data_center from data /
                borderleftstyle=solid
                borderrightstyle=solid
                bordertopstyle=hidden
                borderbottomstyle=hidden;

        style data_bottom from data /
                borderbottomstyle=double;

        style yellowback from data/
                background=yellow;

        style yellowdots from yellowback/
                tagattr="Pattern:Gray125";

    END;

define tagset TAGSETS.CUSTOMEXCELXP;
             parent=tagsets.EXCELXP;
DEFINE EVENT get_borderstyle;
       set $borderstyle borderrightstyle / if cmp( $border_position, "Left");
       set $borderstyle bordertopstyle / if cmp( $border_position, "Right");
       set $borderstyle borderbottomstyle / if cmp( $border_position, "Top");
       set $borderstyle borderleftstyle / if cmp( $border_position, "Bottom");
        set $borderstyle upcase($borderstyle);
        set $borderstyle $borderstyles[$borderstyle];

        do /if ^$borderstyle;
             set $borderstyle "Continuous" /if $borderwidth;
         done;
END;

DEFINE EVENT compile_regexp;
   set $currency_sym "\" $currency;
   set $currency_sym "\$" /if ^$currency_sym;
   set $decimal_separatorsym $decimal_separator;
   set $decimal_separatorsym "\." /if ^$decimal_separatorsym;
   set $decimal_separatorsym "\" $decimal_separator /if cmp( $decimal_separator, ".");
   set $thousands_separatorsym $thousands_separator;
   set $thousands_separatorsym "," /if ^$thousands_separatorsym;
   set $thousands_separatorsym "\" $thousands_separator /if cmp( $thousands_separator,
".");
    set $punctuation $currency $thousands_separator %NRSTR("%%");
    set $integer_re "\d+";
    set $sign_re "[+-]?";
    set $group_re "\d{1,3}(?:" $thousands_separatorsym "\d{3})*";
    set $whole_re "(?:" $group_re "|" $integer_re ")";
    set $exponent_re "[eE]" $sign_re $integer_re;
```

```
    set $fraction_re "(?:" $decimal_separatorsym "\d*)";
    set $real_re "(?:" $whole_re $fraction_re "|" $fraction_re $integer_re "|"
$whole_re ")";
     set $percent_re $sign_re $real_re %NRSTR("\%%");
     set $scinot_re $sign_re "(?:" $real_re $exponent_re "|" $real_re ")";
     set $cents_re "(?:" $decimal_separatorsym "\d\d)";
     set $money_re $sign_re $currency_sym "(?:" $whole_re $cents_re "|" $cents_re "|"
$whole_re ")";
     set $number_re "/^(?:" $real_re "|" $percent_re "|" $scinot_re "|" $money_re
")\Z/";
      eval $number prxparse($number_re);
   set $tagattr_regexp "/^([Ff][Oo][Rr][Mm][Aa][Tt]:|[Ff][Oo][Rr][Mm][Uu][Ll][Aa]:|";
   set $tagattr_regexp $tagattr_regexp "[Rr][Oo][Tt][Aa][Tt][Ee]:|[Tt][Yy][Pp][Ee]:|";
   set $tagattr_regexp $tagattr_regexp "[Hh][Ii][Dd][Dd][Ee][Nn]:|";
   set $tagattr_regexp $tagattr_regexp "[Pp][Aa][Tt][Tt][Ee][Rr][Nn]:|";
   set $tagattr_regexp $tagattr_regexp
"[Mm][Ee][Rr][Gg][Ee][Aa][Cc][Rr][Oo][Ss][Ss]:)/";
   eval $tagattr_regex prxparse($tagattr_regexp);
   eval $cm_re prxparse("/[0-9]*[cC][mM]/");
   eval $in_re prxparse("/[0-9]*[iI][nN]/");
   eval $mm_re prxparse("/[0-9]*[mM][mM]/");
   eval $px_re prxparse("/[0-9]*px/");
   eval $pt_re prxparse("/[0-9]*pt/");
END;

DEFINE EVENT font_interior;

   do /if any( font_face, font_size, font_weight, foreground);
      put "<Font";
      do /if font_face;
         set $fontFace font_face;

         do /if contains( font_face, "Courier");
            set $fontFace tranwrd($fontFace,"sans-serif, ","");
            set $fontFace tranwrd($fontFace,", sans-serif","");
            set $fontFace tranwrd($fontFace,"sans-serif","");
          done;
          set $fontFace tranwrd($fontFace,"SAS Monospace, ","");
          set $fontFace tranwrd($fontFace,"SAS Monospace","");
          set $fontFace tranwrd($fontFace,"'","");
          set $fontFace tranwrd($fontFace," ,",",");
          set $fontFace strip($fontFace);
          set $fontname scan($fontFace,1,",");
          set $fontname strip($fontname);
          eval $count 1;
          unset $tmp_fontFace;
          do /while ^cmp( $fontname, " ");
             stop /if missing($fontname);
             iterate $bad_fonts;
             do /while _value_;
                do /if cmp( $fontname, _value_);
                   unset $fontname /if cmp( $fontname, _value_);
                   stop;
                done;
                next $bad_fonts;
             done;


             do /if $fontname;
                set $tmp_fontFace $tmp_fontFace ", " /if $tmp_fontFace;
                set $tmp_fontFace $tmp_fontFace $fontname;
                unset $fontname;
             done;
```

```
      eval $count $count +1;
      set $fontname scan($fontFace,$count,",");
      set $fontname strip($fontname);
   done;
   set $fontFace $tmp_fontFace;
   eval $comma index($fontFace,",");
   eval $comma_index $comma;
   eval $comma_count 0;
   set $tmp_fontFace $fontFace;

   do /while $comma > 0;
      eval $comma $comma +1;
      eval $comma_count $comma_count +1;
      do /if $comma_count = 3;
         eval $comma_index $comma_index -1;
         set $fontFace substr($fontFace,1,$comma_index);
         stop;
       done;

       set $tmp_fontFace substr($tmp_fontFace,$comma);
       eval $comma index($tmp_fontFace,",");
       eval $comma_index $comma_index + $comma;
    done;

    putq " ss:FontName=" strip($fontFace);
    unset $fontFace;
done;


do /if font_size;

   trigger get_font_height;
    putq " ss:Size=" $font_height;

 else;
    eval $font_height 0;
 done;

 eval $row_height 0;

 do /if $font_height;
    eval $row_height $font_height + $row_height_fudge;
 done;


 do /if $row_height = 0;
    eval $row_height 12 + $row_height_fudge;
 done;


 do /if cmp( htmlclass, "data");
    stop /if $data_point_size;
    set $data_point_size $font_height;
 done;

 set $tmp lowcase(htmlclass);

 do /if contains( $tmp, "systemtitle");
    set $have_title_style "True";

    do /if cmp( $row_heights["Title"], "0");
       set $row_heights["Title" ] $row_height;
```

```
            done;

         done;


         do /if contains( htmlclass, "SystemFooter");
            set $have_footer_style "True";

            do /if cmp( $row_heights["Footer"], "0");
               set $row_heights["Footer" ] $row_height;
            done;
         done;


         do /if cmp( htmlclass, "byline");
             set $have_byline_style "True";

             do /if cmp( $row_heights["Byline"], "0");
                set $row_heights["Byline" ] $row_height;
             done;
          done;


          do /if cmp( htmlclass, "header");
             stop /if $header_point_size;
             set $header_point_size $font_height;
             do /if cmp( $row_heights["Table_head"], "0");
                set $row_heights["Table_head" ] $row_height;
             done;


             do /if cmp( $row_heights["Parskip"], "0");
                set $row_heights["Parskip" ] $row_height;
             done;

           done;

           put " ss:Italic=""1""" /if cmp( FONT_STYLE, "italic");
           put " ss:Bold=""1""" /if cmp( FONT_WEIGHT, "bold");
           putq " ss:Color=" FOREGROUND /if ^cmp( foreground, "transparent");

           do /if text_decoration;
              put $textdecoration["underline" ] /if cmp( text_decoration,
"underline");
              put $textdecoration["strikethrough" ] /if cmp( text_decoration,
"strikethrough");
              put $textdecoration["overline" ] /if cmp( text_decoration, "overline");
              put $textdecoration["blink" ] /if cmp( text_decoration, "blink");
           done;

           put " />" NL;

           do /if $debug_level >= 1;
              putlog "CLASS: " htmlclass;
              iterate $row_heights;
              do /while _name_;
                  putlog _name_ ": " _value_;
                  next $row_heights;
               done;

            done;

         done;
```

```
          do /if cmp( htmlclass, "pagebreak");
              stop /if ^any( background, tagattr, foreground);
              put "<Interior";
              putq " ss:Color=" BACKGROUND;
              putq " ss:Pattern=" tagattr;
              putq " ss:PatternColor=" FOREGROUND /if (tagattr);
              put " />" NL;

           else;

             do /if background;
               put "<Interior";

               do /if ^cmp( background, "transparent");
                   putq " ss:Color=" BACKGROUND;
                   do /if $attrs["pattern"];
                      set $pattern_override propcase($attrs["pattern"]);
                      putq " ss:Pattern=" strip($pattern_override)  /if exist(
BACKGROUND);
                   else;
                     put " ss:Pattern=""Solid""" /if exist( BACKGROUND);
                   done;
                done;

               put " />" NL;

              else /if cmp( htmlclass, "body");
                  put "<Interior ss:Pattern=""Solid"" />" NL;
               done;

           done;

    END;
                map = %NRSTR("<>&|~");
                mapsub = %NRSTR("/&lt;/&gt;/&amp;/&#10;/&#32;/");

       END;

RUN;
quit;

%MEND CREATECUSTOMEXCELXP;


%CreateCustomExcelXP;

ODS TAGSETS.CUSTOMEXCELXP FILE="example.xls" STYLE=styles.slsrep10
OPTIONS (EMBEDDED_TITLES= "YES"   EMBEDDED_FOOTNOTES = "YES" );
ODS TAGSETS.CUSTOMEXCELXP STYLE=styles.slsrep10 options(sheet_name="Example 1");

PROC REPORT DATA=example1 NOWD;
COLUMN txtval bp;
DEFINE txtval / DISPLAY;
DEFINE bp / DISPLAY;

COMPUTE bp;
   IF bp=156 THEN CALL DEFINE("bp","style","STYLE=yellowdots");
ENDCOMP;
RUN;

ODS TAGSETS.CUSTOMEXCELXP STYLE=styles.slsrep10 options(sheet_name="Example 2");
PROC REPORT DATA=example1 NOWD;
```

```
COLUMN txtval bp;
DEFINE txtval/DISPLAY  STYLE(HEADER)=header_right STYLE(COLUMN)=data_left;
DEFINE bp / DISPLAY  STYLE(HEADER)=header_left STYLE(COLUMN)=data_right;
COMPUTE bp;
       IF bp=87 then do;
          CALL DEFINE(_ROW_, "STYLE","STYLE={borderleftwidth=1px borderrightwidth=1px
          borderbottomSTYLE=double borderbottomwidth=1px}");
          CALL DEFINE(_COL_,"STYLE","STYLE={borderrightstyle=double
          borderrightwidth=1px}");
       END;
ENDCOMP;

COMPUTE txtval;
     CALL DEFINE(_COL_,"STYLE","STYLE={borderleftstyle=double borderleftwidth=1px}");
ENDCOMP;
RUN;


ODS TAGSETS.CUSTOMEXCELXP STYLE=styles.slsrep10 options(sheet_name="Example 3");
proc print DATA=example3;
RUN;

ODS TAGSETS.CUSTOMEXCELXP STYLE=styles.slsrep10 options(sheet_name="Example 4");
proc print DATA=example4;
RUN;

ODS tagsets.customexcelcp CLOSE;
```

REFERENCES

- DelGobbo, Vincent. 2009. “More Tips and Tricks for Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®.” *Proceedings of the SAS Global 2009 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings09/152-2009.pdf.

- Gebhart, Eric. 2010. “ODS ExcelXP: Tag Attr Is It! Using and Understanding the TAGATTR= Style Attribute with the ExcelXP Tagset.” *Proceedings of the SAS Global 2010 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings10/031-2010.pdf.

CONTACT INFORMATION

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

Name: Deepak Asrani
Enterprise: Medtronic Inc.
Address: 8200 Coral Sea St NE
City, State ZIP: Mounds View, MN 55112
Work Phone: 763.478.9700
E-mail: deepakgopeasrani@yahoo.com