Paper 187-27

# PROC REPORT: Doin' It in Style!

**Ray Pass, Ray Pass Consulting, Hartsdale, NY**
**Sandy McNeill, SAS, Cary, NC**

## Abstract

The advent of the SAS® Output Delivery System (ODS to its friends) has turned SAS reports from machine-generated, black & white monospace bores into people-produced, productive and reader-friendly information displays. One of the main principles underlying ODS is the use of Table and Style definitions (also known as Table and Style templates). Most procedures have a standard output layout structure and rely on their Table and Style definitions to govern the cosmetic or stylistic appearance of their tables. Certain procedures (REPORT, TABULATE, etc), however, by the very nature of their complete structural customizability, do not rely on fixed external table definitions. For these procedures, stylistic customizations are performed through the use of the STYLE option, an ODS concept which is integrated into the heart of the procedures' syntax. This presentation will demonstrate the use of STYLEs in the REPORT procedure. This Paper will also introduce some of the more advanced structural features of PROC REPORT.

## Introduction

Prior to Version 8 (actually Version 7) of the SAS System, the only form of output available from PROC REPORT was the listing file in the Output Window. The output was produced in SAS Monospace font with form characters (usually dashes) used for overlining and underlining. This was the acceptable (and in fact the *only*) way to bring attention to summary or total lines. There was no way to highlight any of the rows, columns or cells of the output. As Version 8 was released, HTML output was gaining huge popularity as the choice of medium for sharing information – reports, documents, charts. In the HTML world, monospace, fixed fonts were no longer preferred and it was now possible to use proportional fonts, colors, different font sizes, bolding and italics to bring attention to areas of reports that needed more attention from the reader. Version 8 contained the first production release of the Output Delivery System (ODS). One of the main features of ODS is the ability to produce output from all BASE procedures in alternate formats (known in ODS as *destinations*). One of the original ODS destinations, in addition to the default LISTING destination, was HTML. Output sent to this destination was rendered as HTML-tagged output suitable for viewing in HTML browsers.

Most BASE procedures follow certain fairly rigid structural guidelines in terms of the overall layout of the results, and the design of the layout is fairly consistent from run to run. All UNIVARIATE output for example follows a basic blueprint. This was not, however, possible with certain reporting procedures (REPORT, TABULATE, PRINT, etc) because of the infinite amounts of final data layouts that could be created depending on many data factors including variables used and reporting statistics chosen, as well as other design considerations. Therefore, while standard codified aspects of most procedure output could be individually customized via ODS and its accompanying TEMPLATE procedure, this was not possible for the reporting procs because of the lack of standard replicable design features. To compensate for this lack of individual customizability, a system of STYLE formatting was made available for use in REPORT and TABULATE coding (now available in PRINT as well) which provided the ability to individually customize almost all design aspects of the procedure output.

PROC REPORT, while certainly able to mimic the basic reporting output needs of PROC PRINT, can also go way beyond in terms of powerful information display. This presentation will illustrate some of the many different features of ODS STYLEs as implemented in PROC REPORT as well as some of the more advanced, non-ODS tools available in the procedure. This will be done through a series of examples, each using the same source data set. This data set contains values from a fictitious drug trial of an anti-hypertension blood pressure medication. The data collected are from multiple patients and consist of basic demographic information (drug, patient, sex, visit date), along with systolic and diastolic blood pressure readings at time of visit, and reported adverse reactions (fever, nausea, rash). The examples will start off very simply and then build upon each other by adding features until the final example which will be a culmination of these features. Examples 1 and 2 will present a few of the more advanced optional features of PROC REPORT when used with the ODS HTML destination, while 3, 4 and 5 will delve briefly into the use of cosmetic customization with STYLEs. Let's go!

## Example 1

This first example is a simple report which shows demographic data for all patients in the study, their blood pressure readings at time of visit and an indication as to whether or not any adverse reactions were reported. The code for the report is as follows, with comments after the code. The output for all examples can be found at the end of the paper.

```
ods listing close;
ods html body = "Example1.htm";
*--------------------------------------------;
title1 'Blood Pressure Med Study - Vanilla';
*--------------------------------------------;
proc report data=bptrial nowd split='\';

   column patient drug sex visitdate
          systolic diastolic
          fever nausea rash;

   define patient   / order;
   define drug      / order;
   define sex       / order;
   define visitdate / analysis format=date7.;
   define systolic  / analysis;
   define diastolic / analysis;
   define fever     / analysis;
   define nausea    / analysis;
   define rash      / analysis;
```

```
run;
*--------------------------------------------;
ods html close;
ods listing;
```

**Example 1 Code**

The output is sent to the ODS HTML destination via two simple ODS statements:

1) **ODS HTML FILE = 'Example1.htm';** – this statement defines the HTML output file to which the output will be written,

2) **ODS HTML CLOSE;** - this statement closes the output file and is necessary before the output is available for browsing.

The ODS LISTING CLOSE; and ODS LISTING; statements, while not necessary for the functionality of the ODS HTML destination routing, are usually an excellent addition to all ODS coding. They simply turn off, and then turn back on, the default ODS LISTING destination to conserve resources.

Several things should be noticed as you look at the report output: 1) the report is rather long, plain and monotonous, 2) it is not really easy to discern whether a patient had a reaction to the medication, and what the reaction was, 3) it is rather easy to confuse one patient's information with the next since all the patients are listed in one long table. These concerns will be addressed in the following examples.

## Example 2

One of the main problems with the first example is that there is both *too much* information in the one report all thrown together, and *too little* information in terms of no highlighting of important data which should be made intentionally obvious without having the need to hunt for it. These shortcomings are handled by breaking the report into two parts: 1) a summary report listing patient-identifying data as well as a notion of the presence of any adverse reaction at all, 2) a per-patient detail report with each patient encompassing an individually linkable section of the overall report. The most important feature of this example is the creation of the hot-links from the summary report to the individual patient sections of the detail report via a calculated patient sequence number. Here is the code for the reports followed by a discussion of certain aspects of the code:

```
ods listing close;
ods html body = "Example2summ.htm";
*--------------------------------------------;
title1 'Blood Pressure Med Study - Summary
(basic)';
*--------------------------------------------;
proc report data=bptrial nowd split='\';

   column patient drug sex fever nausea
          rash reaction;

   define patient  / group;
   define drug     / group;
   define sex      / group;
   define fever    / analysis sum noprint;
   define nausea   / analysis sum noprint;
   define rash     / analysis sum noprint;
   define reaction / computed 'Reaction?';

   compute reaction / length=3;
      if   fever.sum  = .
      and  nausea.sum = .
      and  rash.sum   = . then reaction = 'No ';
                          else reaction = 'Yes';
   endcomp;

   compute before patient;
      ptno + 1;
   endcomp;

   compute patient;
      urlstring = "Example2.htm#pt"
                  || left(put(ptno,3.0));
      call define (_col_,'url',urlstring);
   endcomp;
run;
*--------------------------------------------;
ods html close;
*--------------------------------------------;
ods html body   = "Example2.htm"
         anchor = "pt1";
*--------------------------------------------;
title1 'Blood Pressure Med Study - Detail
(basic)';
*--------------------------------------------;
proc report data=bptrial nowd split='\';

   column patient drug sex visitdate
          ("Blood Pressure"
           systolic slash diastolic)
          fever nausea rash;

   define patient   / order;
   define drug      / order;
   define sex       / order;
   define visitdate / analysis format=date7.;
   define systolic  / analysis 'Systolic';
   define slash     / computed '/';
   define diastolic / analysis 'Diastolic' left;
   define fever     / analysis center;
   define nausea    / analysis center;
   define rash      / analysis center;

   break after patient / page;

   compute slash / length=1;
      slash = '/';
   endcomp;
run;
*--------------------------------------------;
ods html close;
ods listing;
```

**Example 2 Code**

Let's talk about the two REPORT procs separately and then see how they are linked. First the summary report. As each new patient is processed in the code, a sequential patient counter variable is created (PTNO). Note that a *DATA step* variable (not included in the COLUMN statement and therefore not re-initialized with each new output record) is used rather than a *report* variable which would always be reset to 0 and would therefore not increment. The next part of the process is the computation of another new *DATA step* variable, URLSTRING. This is done in a COMPUTE block associated with the PATIENT variable. Note that you can use COMPUTE blocks on non-COMPUTED variables. URLSTRING is a concatenation of the name of the *detail* output file "Example2.htm", the characters "pt#", and the current value of PTNO. So, for the first patient, URLSTRING has a value of "Example2.htm#pt1", and so on. This last part of the string, "#pt1", is called an *anchor*, and identifies a specific location in the HTML file "Example2.htm".

The other thing we do in the PATIENT COMPUTE block is to use the CALL DEFINE statement to associate some action with the current value (display) of PATIENT. There are many things you can do with a CALL DEFINE statement; in this case we are turning the displayed value of PATIENT into a hot link. CALL DEFINE takes three parameters. In this case the first one tells it what to operate on - the current column (_COL_). The second parameter tells it what to do - create a 'URL' hot link, and the third parameter gives a value for the hot link, namely the current value of the data step variable URLSTRING.

Now let's jump to the detail report. The first thing added is the constant COMPUTED variable SLASH. A simple addition, but one that serves to make the blood pressure readings more familiar looking. We also CENTER the displays of the reaction variables.

Now the fun stuff begins. Notice that we are using a BREAK AFTER PATIENT / PAGE; statement. What this statement would normally do in the LISTING destination file would be to go to a new page whenever the value of the variable PATIENT changed, i.e, with each new patient identification number. However, since HTML does not have the concept of physical pages, REPORT uses the statement as an indication to start a new table in the HTML output for each patient. When multiple tables are created within one HTML file, each table is automatically assigned an identifying anchor string which appears in the HTML code and can be referenced. The default values of these anchor strings are "IDXnumber", with "number" usually starting from 0 and incrementing by 1 for each new table. We can, however, change the base of the anchor from "IDX" to any other string, and the initial sequential value to any starting number. This is done with the ANCHOR option on the ODS HTML statement. In our example, we set the anchor base/starting number to "pt1". So the first patient table will have an anchor string of "pt1", the second "pt2", and so on, therefore giving each table in the detail report file a unique identifying string. This string is what is used in the summary table to hot-link the patient number display line directly to the patient detail table inside the detail report file.

The summary and detail tables are therefore tied together by the creation of anchor points in the detail output and corresponding hot links in the summary table which point to the anchors in the detail table. For example, the first patient's patient number, 1813, is displayed as 1813 in the summary table. This is defined via the CALL DEFINE statement as a hot-link to a specific location in the detail table, namely "Example2#pt1", the beginning of the detail display for patient 1813. Pretty neat.

## Example 3

OK, now that we know how to link lines in a summary report to corresponding portions of a detail report, let's work on making the detail reports more informative. We're not going to change any of the data presented, but we *are* going to make them more useful by making them easier to absorb at a glance. Cosmetics are really important in the world of information transmission. We are going to be using color and pictures (gifs), and by doing so in a data dynamic manner, the reports will become much more pleasing to the eye. This will in turn make it much easier to identify trends in the data.

Examples 3 and 4 deal solely with the detail reports and use the exact same summary report created in Example 2 as a jumping off platform for access to the detail reports. Here is the code for Example 3 followed by discussion:

```
ods listing close;
ods html body   = "Example3.htm"
         anchor = "pt1";
*-------------------------------------------;
title1 'Blood Pressure Med Study - Detail
(enhanced-1)';
title2 'Add Images and Data Above Headers';
*-------------------------------------------;
proc report data=bptrial nowd split='\'
            style=[preimage="medical.gif"];

   column patient drug sex visitdate
          ("Blood Pressure"
           systolic slash diastolic)
          fever nausea rash;

   define patient   / order noprint;
   define drug      / order;
   define sex       / order;
   define visitdate / analysis format=date7.;
   define systolic  / analysis 'Systolic';
   define slash     / computed '/';
   define diastolic / analysis 'Diastolic' left;
   define fever     / analysis center;
   define nausea    / analysis center;
   define rash      / analysis center;

   compute before patient;
      if   fever.sum  = .
      and  nausea.sum = .
      and  rash.sum   = . then reaction = 'No ';
                           else reaction = 'Yes';

      lastsys  = .;
      lastdias = .;
   endcomp;


   compute before _page_ / center;
    line 'Patient Number: '        patient  $5.;
    line 'Reaction to medicine?: ' reaction $3.;
   endcomp;

   compute slash / length=1;
      slash = '/';
   endcomp;

   compute systolic;
      if lastsys ne . then do;
         if systolic.sum gt lastsys
         then call define (_col_,'style',
           'style=[foreground=red
                    preimage="trendupsm.gif"]');
         else if systolic.sum lt lastsys
         then call define (_col_,'style',
           'style=[foreground=green
                   preimage="trenddownsm.gif"]');
      end;
      lastsys = systolic.sum;
   endcomp;

   compute diastolic;
      if lastdias ne . then do;
         if diastolic.sum gt lastdias
         then call define (_col_,'style',
           'style=[foreground=red
                    postimage="trendupsm.gif"]');
         else if diastolic.sum lt lastdias
         then call define (_col_,'style',
           'style=[foreground=green
                  postimage="trenddownsm.gif"]');
```

```
      end;
      lastdias = diastolic.sum;
   endcomp;

   break after patient / page;
run;
*-----------------------------------------------;
ods html close;
ods listing;
```

**Example 3 Code**

As you look at the detail report for this example, the first thing that catches your eye is the graphic that precedes each of the patients' reports. In this paper we just used one of Micosoft's collection of "gifs", but this very well could be your company's logo placed before each of the tables (we could also have placed only one gif before the entire collection of patients). This graphic is called a "preimage" because it comes before each table. We could also have placed it, or an additional image, after each table, in which case it would be a "postimage". In this example, we placed the image before each table by including it as an attribute in the STYLE option on the PROC REPORT statement as follows:

```
STYLE=[PREIMAGE="medical.gif"]
```

The STYLE option can be used in a PROC REPORT statement, a DEFINE statement, a CALL DEFINE statement, a BREAK or RBREAK statement, or a COMPUTE statement. Using the STYLE option in a PROC REPORT statement allows you to set up certain style properties to be used for an entire report. Using it in any of the other statements in which it is allowed, creates style properties for the portion(s) of the report governed by the particular statement. The basic syntax of the STYLE option is:

```
STYLE(location)=style-element-name
      [attribute1=attribute1-value
       attribute2=attribute2-value
       .     .     .
       attributeN=attributeN-value}
```

The optional (location) tells PROC REPORT where in the report the style should be applied. Possible values are REPORT, COLUMN, HEADER, LINES, CALLDEF and SUMMARY. Since the location is optional, you can very often leave it out (as we have in this example), with the default location being assigned depending upon which statement the STYLE option is being used in. The default location for the REPORT statement is REPORT which in this case is what we want since we want our STYLE option to be applied to each of the reports that is created. The style-element-name refers to the name of a style element from a style template that is being used, if one is being used. This is out of the scope of discussion of this paper, but if you want to learn more about style elements, either the Online or the hardcopy documentation for V8 is the place to go.

The last part(s) of the STYLE option are pairings of the attribute names for those style attributes that you want to set, and the values that you want to set them to. The PROC REPORT documentation contains complete lists of attributes and their possible values. In our example, we have chosen the medical image contained in the file "medical.gif" to appear before each report by setting the PREIMAGE attribute value to "medical.gif". The absence of a full path indicates that the file is to be found in the same location as the program.

Now let's go back and take another look at the report in this example. Starting at the top of the actual data output, notice that we have added overall patient header material *before* the column headers. This was not possible prior to V8, but is now easily doable with the new _PAGE_ location for the COMPUTE statement. The actual text displayed comes from the LINE statements in the first COMPUTE block and the derived REACTION variable.

Notice that there are images in some of the actual data cells. These images correspond to how the patient's blood pressures compared to the values on the previous visit, and they are visually and color-coded. An increased value since the last visit is shown in red with a red upward arrow, and a decreased value is shown in green with a green downward arrow. This change of value from visit to visit is an important part of the content of this report and is now made visually compelling. The data in this report are the same as in Example 2, but the changes in values are much more apparent with the images and colors in this example.

So how did we do that? First of all, since we are reporting on two separate blood pressure measures, SYSTOLIC and DIASTOLIC, we will need to work on the two columns separately, but we'll only describe the processing for one of them here, SYSTOLIC. Processing for DIASTOLIC is exactly the same as for SYSTOLIC. Since we want to make stylistic changes to the values in the SYSTOLIC column, we need to have a COMPUTE block for that column. This COMPUTE block will execute for each row each time the SYSTOLIC column is being processed. The processing that must take place in the COMPUTE block is a comparison of the current pressure value to the one for the previous record (visit), so we need to set up a variable with the last value so that we have something to compare against. In the COMPUTE BEFORE PATIENT block, we initialize LASTSYS to missing. Since LASTSYS is a *DATA step* variable, it will not automatically re-initialize to missing with each new record *until* a new patient is encountered. In the COMPUTE SYSTOLIC block, the first thing we do is check the value of LASTSYS. If it is missing, as it will be on the first visit record for each patient, the IF loop does not execute and we simply set the value of LASTSYS to the current value of SYSTOLIC. If it is *not* missing, we have a value to compare the current value of SYSTOLIC against, and we enter into the comparison section. After the comparison work is done, we set LASTSYS to the current value in preparation for the next record. Notice in the COMPUTE block that there are two conditional CALL DEFINE statements. One will be executed if the current systolic value is greater than LASTSYS and the other one will be executed if the current systolic value is less than LASTSYS. The two CALL DEFINE statements are essentially identical with the only difference between the two being the image and the color being set. Let's look at one of these CALL DEFINE statements:

```
CALL DEFINE(_COL_, 'STYLE',
 'STYLE=[FOREGROUND=RED
         PREIMAGE="trendupsm.gif"]');
```

Just as we saw in Example 2, this CALL DEFINE is using the _COL_ location tag which means that whatever is set from this CALL DEFINE will be applied to the current column (SYSTOLIC in this instance since we are in the COMPUTE block for the SYSTOLIC column) on the current row (or in other words, the current cell). The second parameter, however, is different. This time we are using 'STYLE' as our action tag

(in Example 2 we used 'URL' to create a hot-link). This indicates to PROC REPORT that we are going to be setting STYLE attribute(s), which will be found in the third parameter. Note that 'STYLE' is surrounded with quotes here because it is really a parameter to the CALL DEFINE statement. The syntax of the STYLE attribute/value pairings is the same as we saw before. This time we have two attributes that we are setting in this STYLE option, so we have two pairings of attributes and their values.

Let's look at the first CALL DEFINE, the one that gets executed if the current value of SYSTOLIC is greater than LASTSYS. We choose to denote this occurrence by setting the color of the text in the cell to be red and by inserting an upward arrow image before the text in the cell. The color of the text is referred to as the FOREGROUND color (we could have also changed the BACKGROUND color of the cell if we had chosen to do so). Since we want the text to be red, we simply code FOREGROUND=RED as the first attribute/value pairing. The second attribute/value pairing looks very similar to the one we use in our STYLE option in the PROC REPORT statement since again we are using the PREIMAGE attribute. This brings up an important point: the same attribute name used in a different statement influences a different area of the report. Since this PREIMAGE attribute is being used in a CALL DEFINE statement which is using the _COL_ location tag, the effect of the PREIMAGE attribute here will be to place the image inside the current cell before the text that will appear in the cell. Notice that in the DIASTOLIC column, we placed the image *after* the text in the cell by using a POSTIMAGE attribute instead. This allowed us to keep the actual numbers close to each other, separated by the slash. The image that we place in any cell in which the systolic value is greater than the last systolic value is a trend up small image (TRENDUPSM.GIF). This gives us our second attribute/ value pairing, PREIMAGE= "trendupsm.gif".

A glance at the CALL DEFINE statement when the current value of SYSTOLIC is less than the last SYSTOLIC value shows just two differences: the color value is GREEN instead of RED and the image used is a trend down small image instead of an up image. Also, since SYSTOLIC and DIASTOLIC are separated into two different columns, we use basically the same logic in a COMPUTE block for DIASTOLIC.

## Example 4

This time, we are going to once again just concentrate on the detail reports. Example 4 builds upon what we just completed for Example 3, and adds some visual enhancements as well as some patient summary data (mean pressure values.) Take a look at the output for Example 4. Now those are nice looking tables! What's different compared to the Example 3 output? The line statements before the column headings now have color added. The use of more color elsewhere, especially in the adverse reaction cells, also adds a great deal. And that really is the point – use color, font, bolding, italicizing, and other stylistic attributes to emphasize different areas of your reports to significantly enhance their informational impact.

OK, cosmetics are really important, but we've also added some data to the top portion of each patient's output. We now show the mean values of SYSTOLIC and DIASTOLIC over all visits for the patient. We'll get to how we did this below.

As you look through the code for Example 4, you'll notice that only small amounts of code have changed compared to Example 3. We'll start from the top of the code and work our way down to see what is different. But first, the code:

```
ods listing close;
ods html body   = "Example4.htm"
         anchor = "pt1";
*-------------------------------------------;
title1 'Blood Pressure Med Study - Detail
(enhanced-2)';
title2 'Add Means and Background Colors
(including reactions)';
*-------------------------------------------;
proc report data=bptrial nowd split='\'
          style(report)={preimage="medical.gif"
                         background=red}
          style(header)={background=lightskyblue
                         foreground=black}
          style(column)={background=lightcyan
                         foreground=black};

   column patient drug sex visitdate
          ("Blood Pressure"
            systolic slash diastolic)
          ("Reactions" fever nausea rash)
          systolic=sysmean diastolic=diasmean;

   define patient   / order noprint;
   define drug      / order;
   define sex       / order;
   define visitdate / analysis format=date7.;
   define systolic  / analysis 'Systolic';
   define slash     / computed '/';
   define diastolic / analysis 'Diastolic' left;
   define fever     / analysis center;
   define nausea    / analysis center;
   define rash      / analysis center;
   define sysmean   / mean noprint;
   define diasmean  / mean noprint;

   compute before patient;
      if  fever.sum  = .
      and nausea.sum = .
      and rash.sum   = . then reaction = 'No ';
                          else reaction = 'Yes';

      lastsys  = .;
      lastdias = .;
   endcomp;

   compute before _page_ / center
                style=[font_weight=bold
                       foreground=black
                       background=lightyellow];
    line 'Patient Number:  '       patient  $5.;
    line 'Reaction to medicine?  ' reaction $3.;
    line 'Systolic mean: '         sysmean   3.;
    line 'Diastolic mean: '        diasmean  3.;
   endcomp;

   compute slash / length=1;
      slash = '/';
   endcomp;

   compute systolic;
      if lastsys ne . then do;
         if systolic.sum gt lastsys
         then call define(_col_,'style',
                   'style=[foreground=red
                   preimage="trendupsm.gif"]');
         else if systolic.sum lt lastsys
         then call define(_col_,'style',
                'style=[foreground=green
```

```
                  preimage="trenddownsm.gif"]');
      end;
      lastsys = systolic.sum;
   endcomp;

   compute diastolic;
      if lastdias ne . then do;
         if diastolic.sum gt lastdias
         then call define(_col_,'style',
                     'style=[foreground=red
                  postimage="trendupsm.gif"]');
         else if diastolic.sum lt lastdias
         then call define(_col_,'style',
                     'style=[foreground=green
                 postimage="trenddownsm.gif"]');
      end;
      lastdias = diastolic.sum;
   endcomp;

   compute fever;
      if fever.sum ne . then
         call define(_col_,'style',
                     'style=[background=pink]');
   endcomp;

   compute nausea;
      if nausea.sum ne . then
         call define(_col_,'style',
                  'style=[background=#d8859f]');
   endcomp;

   compute rash;
      if rash.sum ne . then
         call define(_col_,'style',
                  'style=[background=#bb2222]');
   endcomp;

   break after patient / page;
run;
*-----------------------------------------------;
ods html close;
ods listing;
```

**Example 4 Code**

The first difference is in the STYLE option, or rather option*s*, in the PROC REPORT statement. In Example 3, we only had one STYLE option in that statement – now we have three. As we noted earlier, a STYLE option used in a PROC REPORT statement without a LOCATION defaults to a location value of REPORT. Since we are now using multiple locations, we denote each one explicitly, as in STYLE(REPORT), etc. We are now also using multiple attribute pairings in each location. When the location used is REPORT, the attributes named will be applied to the entire report. When we use the BACKGROUND attribute at this location, it colors the entire background of the report. However, since the cells themselves have a default background color applied (coming from the STYLE template), the effect is that the only part of the background of the report that is left showing through that is not overwritten are the lines separating the cells and the border line of the entire table (the grid of the table). So setting the BACKGROUND for the REPORT has the effect of coloring the grid of the table. We use the HEADER location to color the background of the column header area one shade of blue, and the COLUMN location to color the background of all the column cells another shade of blue. Both areas have their foregrounds (text characters) set to black. It will take a bit of work to learn exactly which STYLE locations control which output areas, but it will pay off.

While we are at the top of the report, note that we are also using a STYLE option on the COMPUTE BEFORE _PAGE_ block. The STYLE option for this COMPUTE block has three attribute/value pairings: FONT_WEIGHT = BOLD, FOREGROUND = BLACK, and BACKGROUND = LIGHTYELLOW. As noted above, the PROC REPORT documentation lists all the possible STYLE attributes and their possible values.

The next item that's new in this example is the addition of those mean blood pressure values up top. We accomplish this by first creating aliases for SYSTOLIC and DIASTOLIC in the COLUMN statement, namely SYSMEAN and DIASMEAN. We choose MEAN as the statistic to use for these aliases in their respective DEFINE statements, and then just add them as LINE variables in the COMPUTE BEFORE _PAGE_ statement. Notice that we are not printing them on each data output line, via NOPRINT options, but we need them in the COLUMN statement or else they will not be available for the COMPUTE block. Easy enough.

The next main code difference that we notice is that there are now three more COMPUTE blocks, one for each of the three different types of adverse reactions. In Example 3, we were already placing an 'X' in a specific reaction column if that particular reaction was reported for that visit. If we look at Patient 1878 for example, we see he reported a reaction of NAUSEA on the 15JAN91 visit, FEVER on the 22JAN91 visit, and a RASH on the 05FEB91 visit. If we were paging down rather rapidly through the tables in this detail report file, we might not have noticed any of those reactions. The purpose of these new COMPUTE blocks is to color code those reaction cells which have an 'X' value so that they are more apparent to the reader's attention.

Since the three COMPUTE blocks for FEVER, NAUSEA, and RASH are essentially identical with the only difference being the color that will be set for the BACKGROUND of the cell, we'll look at only one of them, the FEVER COMPUTE block. If the current value for FEVER is not missing (which means it has a value), then a CALL DEFINE statement is invoked which uses the STYLE action tag as we saw in Example 3, with one attribute/ value pairing: BACKGROUND=PINK. The BACKGROUND attribute sets the background color of the cell – in this case the color pink. For NAUSEA and RASH, we are using hex-encoded RBG (red, blue, green) values. For NAUSEA, we use # D8859F which is a deeper pink, like rose, and for RASH, we use #BB2222, a much deeper hue. There are actually several different ways to specify a color value, two of which we have seen here. Another valid way is to specify the value using GRAPH color notation, like CXBB2222 for our rose colored rash.

## Example 5

In this last example, we will demonstrate another way to achieve the cell highlighting in the columns for FEVER, NAUSEA, and RASH. We'll also add some color to the summary report, and create a hot link to navigate back to the summary report from the details. Here's the code:

```
ods listing close;
*-----------------------------------------------;
proc format;
   value fevfmt  1 = 'pink'
                 . = 'lightcyan';
```

```
   value nausfmt 1 = '#d8859f'
                 . = 'lightcyan';
   value rashfmt 1 = '#bb2222'
                 . = 'lightcyan';
run;
*--------------------------------------------;
ods html body = "Example5summ.htm";
*--------------------------------------------;
title1 "Blood Pressure Med Study - Summary
(enhanced)";
title2 "Add Colors";
*--------------------------------------------;
proc report data=work.bptrial nowd
          style(report)=[rules=all
                         cellspacing=0
                         bordercolor=gray]
          style(header)=[background=lightskyblue
                         foreground=black]
          style(column)=[background=lightcyan
                         foreground=black];

   column patient drug sex fever nausea rash
          reaction;

   define patient  / group;
   define drug     / group;
   define sex      / group;
   define fever    / analysis sum noprint;
   define nausea   / analysis sum noprint;
   define rash     / analysis sum noprint;
   define reaction / computed 'Reaction?';

   compute reaction / length=3;
      if  fever.sum  = .
      and nausea.sum = .
      and rash.sum   = . then reaction = 'No ';
      else do;
         reaction = 'Yes';
         call define(_col_,'style',
                     'style=[foreground=red
                      background=pink
                      font_weight=bold]');
      end;
   endcomp;

   compute before patient;
      ptno + 1;
   endcomp;

   compute patient;
      urlstring = "Example5.htm#pt"
                  || left(put(ptno,3.0));
      call define (_col_, 'url', urlstring );
   endcomp;
run;
*--------------------------------------------;
ods html close;
*--------------------------------------------;
ods html body   = "Example5.htm"
         anchor = "pt1";
*--------------------------------------------;
title1 "Blood Pressure Med Study - Detail
(enhanced-3)";
title2 "Alternate Method for Reaction Colors &
Navigation Title";
title3 "<H4><A HREF='&repout.rephow5summ.htm'>
All Patient Summary</A></H4>";
*--------------------------------------------;
proc report data=work.bptrial nowd split='\'
          style(report)=[rules=all
                         cellspacing=0
                         bordercolor=gray]
          style(header)=[background=lightskyblue
                         foreground=black]
          style(column)=[background=lightcyan
                         foreground=black];

   column patient drug sex visitdate
          ("Blood Pressure"
            systolic slash diastolic)
          ("Reactions" fever nausea rash)
          systolic=sysmean diastolic=diasmean;

   define patient   / order noprint;
   define drug      / order;
   define sex       / order;
   define visitdate / analysis format=date7.;
   define systolic  / analysis 'Systolic';
   define slash     / computed '/';
   define diastolic / analysis 'Diastolic' left;
   define fever     / analysis sum center
          style(column)=[background=fevfmt.];
   define nausea    / analysis sum center
          style(column)=[background=nausfmt.];
   define rash      / analysis sum center
          style(column)=[background=rashfmt.];
   define sysmean   / mean noprint;
   define diasmean  / mean noprint;

   compute before _page_ / center
                style=[font_weight=bold
                       foreground=black
                       background=lightyellow];
     line 'Patient Number: '       patient  $5.;
     line 'Reaction to medicine? ' reaction $3.;
     line 'Systolic mean: '        sysmean   3.;
     line 'Diastolic mean: '       diasmean  3.;
   endcomp;

   compute before patient;
      if  fever.sum  = .
      and nausea.sum = .
      and rash.sum   = . then reaction = 'No ';
                          else reaction = 'Yes';

      lastsys  = .;
      lastdias = .;
   endcomp;

   compute slash / length=1;
      slash = '/';
   endcomp;

   compute systolic;
      if lastsys ne . then do;
         if systolic.sum gt lastsys
         then call define(_col_,'style',
                     'style=[foreground=red
                             font_weight=bold
              preimage="&gifs.trendupsm.gif"]');
         else if systolic.sum lt lastsys
         then call define(_col_,'style',
                     'style=[foreground=green
                             font_weight=bold
            preimage="&gifs.trenddownsm.gif"]');
      end;
      lastsys = systolic.sum;
   endcomp;

   compute diastolic;
      if lastdias ne . then do;
         if diastolic.sum gt lastdias
         then call define(_col_,'style',
                     'style=[foreground=red
                             font_weight=bold
             postimage="&gifs.trendupsm.gif"]');
         else if diastolic.sum lt lastdias
         then call define(_col_,'style',
                     'style=[foreground=green
                             font_weight=bold
```

```
            postimage="&gifs.trenddownsm.gif"]');
      end;
      lastdias = diastolic.sum;
   endcomp;

   break after patient / page;
run;
*-----------------------------------------------;
ods html close;
ods listing;
```

**Example 5 Code**

In Example 4, we used COMPUTE blocks for coloring the cells in the FEVER, NAUSEA, and RASH columns if any of these reactions were present. The DATA step code in these blocks determined if there was a value in the current cell, and if so, it executed a CALL DEFINE statement which used a STYLE parameter to set the BACKGROUND color attribute for that cell. In this example a different method is used to accomplish this result - a user-defined format is created for each of the reaction columns, with the format defining the color for each cell in the column. The three formats that we have created at the top of the code, FEVFMT., NAUSFMT. and RASHFMT, are used to provide the needed background cell color via STYLE options on the DEFINE statements for the variables FEVER, NAUSEA and RASH. Notice that the formatted value if the cell value is 1 is the same color value that was used for the background attribute in the CALL DEFINE statement in Example 4. The other color value in the user-defined formats, 'LIGHTCYAN', is the default background color of the cell as set in the STYLE(COLUMN) option on the PROC REPORT statement. When we use user-defined formats to assign the background color for a cell, we must account for any value that might be in the cell. If we do not have a corresponding formatted value for any cell's data value, the background color will be undefined and we will get unpredictable results. Since there can only be two values in the cells for these columns, missing or 1, we have listed these two values when defining the format.

So, instead of using a STYLE option in a CALL DEFINE statement in a COMPUTE block, we are now using a STYLE option on the DEFINE statement for each of these columns. And instead of using a hard-coded color string as the attribute value for the BACKGROUND attribute, we use the user-defined format name for each variable. Notice also that on the STYLE option in the DEFINE statement, we are using the COLUMN location. This is because the default STYLE locations for a DEFINE statement are both COLUMN and HEADER, and here we only want this style applied to the column (or cell) values, and not the column header value.

The last method used here that we want to talk about is that hot link above each detail report which can bring us back to the summary report. This is accomplished by simply adding a TITLE line to the detail report, which includes an HMTL HREF tag in it, and pointing the hot link to the summary report.

So now in typical SAS fashion, we have seen more than one way to get the job done, in this case the job being the highlighting of cells in PROC REPORT columns. But there is more to this example than just a different coding style; there is also a performance increase by being able to delete those three COMPUTE blocks. Remember that *each* COMPUTE block is executed for *each* row *each* time the column is processed. This means that for these three columns, there are three COMPUTE blocks executed for each row of this report. Deleting these COMPUTE blocks means both a savings of time and memory. Here are some numbers from Version 8.2 executing on a laptop PC. Are these significant savings? Could be, maybe not. The point is that there is always another way, maybe even a better way.

Example 4:

```
NOTE: There were 234 observations read from the
data set WORK.BPTRIAL.
NOTE: PROCEDURE REPORT used:
      real time           0.98 seconds
      user cpu time       0.69 seconds
      system cpu time     0.06 seconds
      Memory                           1437k
```

Example 5:

```
NOTE: There were 234 observations read from the
data set WORK.BPTRIAL.
NOTE: PROCEDURE REPORT used:
      real time           0.86 seconds
      user cpu time       0.63 seconds
      system cpu time     0.04 seconds
      Memory                           1160k
```

## Conclusion

Well we all know that PROC REPORT is a very powerful report generating tool that we've had use of in a series of generations since the last century (very early 1990's that is.) And, we all know how much ODS has improved the presentation and delivery of our SAS productions. Given these powerful tools, it behooves us to use them to their utmost. There are many extensions of basic PROC REPORT coding which can be used to greatly enhance its information delivery prowess. This paper has attempted to demonstrate only a few of them. Reports can easily be made somewhat interlinkable through the use of CALL DEFINE 'URL' parameters and HTML coding. The STYLE option can be used in *many* different locations in PROC REPORT code to greatly enhance the presentation quality, and therefore the meaningful information usability of the reports. These are only two of the many "advanced" features of PROC REPORT. They are really only just jumping off points for the enterprising report creator.

## Acknowledgements

SAS is a registered trademark of the SAS Institute Inc., Cary, NC, USA.

## Author Contact Information

The authors of this paper can be contacted as follows:

**Ray Pass**
Ray Pass Consulting
5 Sinclair Place
Hartsdale, NY 10530
Voice: (914) 693-5553
Fax: (914) 206-3780
e-mail: raypass@att.net

**Sandy McNeill**
SAS Institute
SAS Campus Drive
Cary, NC 27513
Voice: (919) 531-5453
Fax: (919) 677-4444
e-mail: sandy.mcneill@sas.com

Blood Pressure Med Study - Vanilla

| patient | drug | sex | visitdate | systolic | diastolic | fever | nausea | rash |
|---|---|---|---|---|---|---|---|---|
| 1813 | A | F | 08JAN91 | 109 | 73 | | | |
| | | | 15JAN91 | 109 | 73 | | | |
| | | | 22JAN91 | 107 | 73 | | | |
| | | | 29JAN91 | 109 | 73 | | | |
| | | | 05FEB91 | 100 | 74 | | | |
| | | | 12FEB91 | 114 | 73 | | | |
| 1826 | B | F | 09JAN91 | 125 | 89 | | | |
| | | | 16JAN91 | 124 | 89 | | | |
| | | | 23JAN91 | 123 | 86 | | | |
| | | | 30JAN91 | 122 | 86 | | | |
| | | | 06FEB91 | 120 | 90 | X | | |
| | | | 13FEB91 | 119 | 89 | | X | |
| 1839 | B | M | 10JAN91 | 149 | 108 | | | |
| | | | 17JAN91 | 147 | 108 | | | |
| | | | 24JAN91 | 144 | 110 | | | |
| | | | 31JAN91 | 141 | 108 | | | |
| | | | 07FEB91 | 138 | 109 | | | |
| | | | 05FEB91 | 110 | 71 | | X | X |
| | | | 12FEB91 | 100 | 74 | | | |
| 2281 | A | F | 09JAN91 | 140 | 63 | X | | X |
| | | | 16JAN91 | 139 | 63 | X | X | |
| | | | 23JAN91 | 138 | 61 | | | X |
| | | | 30JAN91 | 136 | 61 | | | |
| | | | 06FEB91 | 135 | 64 | X | X | |
| | | | 13FEB91 | 133 | 62 | X | X | X |
| 2294 | A | M | 10JAN91 | 122 | 96 | X | X | |
| | | | 17JAN91 | 128 | 96 | X | X | |
| | | | 24JAN91 | 116 | 94 | | X | X |
| | | | 31JAN91 | 135 | 97 | X | | |
| | | | 07FEB91 | 119 | 95 | X | X | X |
| | | | 14FEB91 | 112 | 95 | | X | |
| 2307 | B | M | 11JAN91 | 101 | 80 | | X | X |
| | | | 18JAN91 | 105 | 80 | | | |
| | | | 25JAN91 | 119 | 78 | X | X | |
| | | | 01FEB91 | 109 | 80 | | | |
| | | | 08FEB91 | 103 | 80 | X | X | X |
| | | | 15FEB91 | 100 | 82 | X | X | |

**Figure 1 – Example 1, Partial HTML Output**

Blood Pressure Med Study - Summary (basic)

| patient | drug | sex | Reaction? |
|---|---|---|---|
| 1813 | A | F | No |
| 1826 | B | F | Yes |
| 1839 | B | M | No |
| 1852 | B | F | Yes |
| 1865 | B | F | No |
| 1878 | A | F | Yes |
| 1891 | B | M | No |
| 1904 | A | M | No |
| 1917 | B | M | Yes |
| 1930 | A | M | Yes |
| 1943 | B | F | Yes |
| 1956 | B | M | Yes |
| 1969 | B | F | No |

**Figure 2a – Example 2, Partial HTML Output – Summary Report**

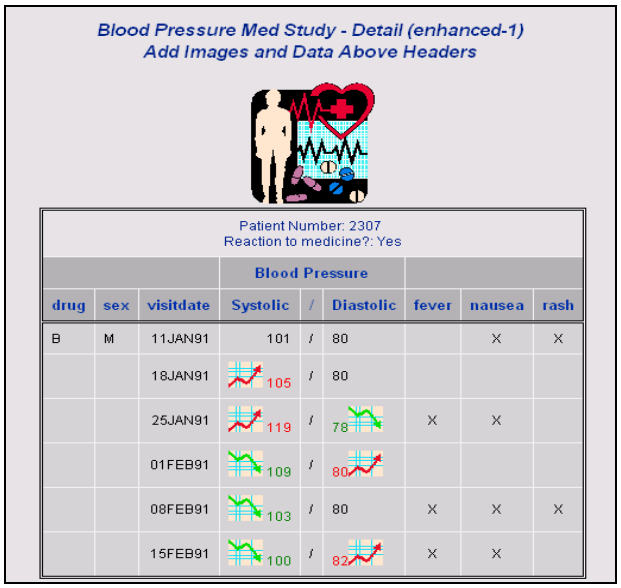
Blood Pressure Med Study - Detail (basic)

| | | | | Blood Pressure | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| patient | drug | sex | visitdate | Systolic | / | Diastolic | fever | nausea | rash |
| 2086 | A | M | 09JAN91 | 148 | / | 79 | | | |
| | | | 16JAN91 | 147 | / | 76 | | X | |
| | | | 23JAN91 | 146 | / | 78 | X | X | |
| | | | 30JAN91 | 144 | / | 80 | | | |
| | | | 06FEB91 | 143 | / | 75 | | X | X |
| | | | 13FEB91 | 141 | / | 73 | | X | |

Blood Pressure Med Study - Detail (basic)

| | | | | Blood Pressure | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| patient | drug | sex | visitdate | Systolic | / | Diastolic | fever | nausea | rash |
| 2099 | B | F | 10JAN91 | 118 | / | 77 | | X | X |
| | | | 17JAN91 | 112 | / | 77 | | | |
| | | | 24JAN91 | 117 | / | 78 | | | |
| | | | 31JAN91 | 123 | / | 77 | | | |

**Figure 2b – Example 2, Partial HTML Output – Detail Report**

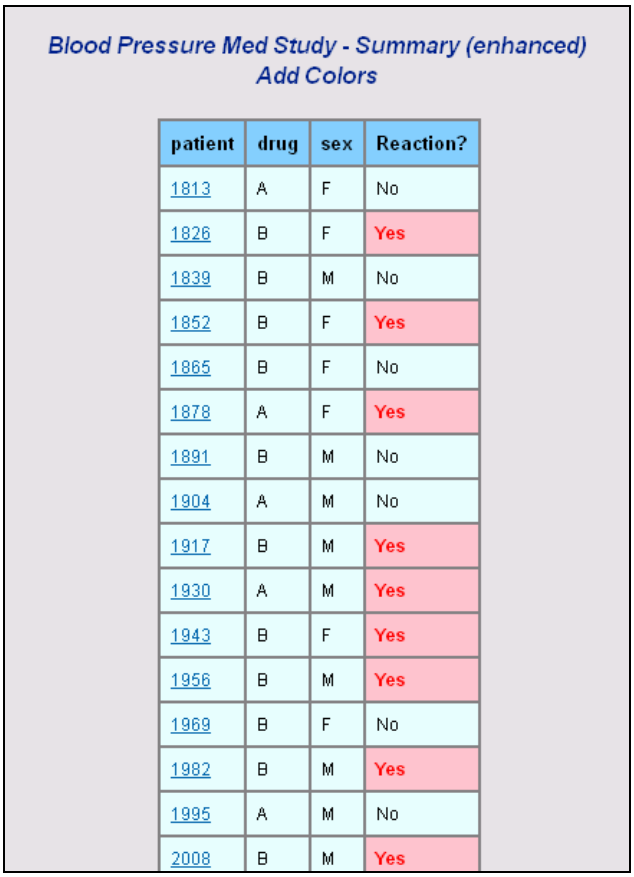


Blood Pressure Med Study - Detail (enhanced-1)
Add Images and Data Above Headers

Patient Number: 2307
Reaction to medicine?: Yes

| | | | Blood Pressure | | | | | |
|---|---|---|---|---|---|---|---|---|
| drug | sex | visitdate | Systolic | / | Diastolic | fever | nausea | rash |
| B | M | 11JAN91 | 101 | / | 80 | | X | X |
| | | 18JAN91 | 105 | / | 80 | | | |
| | | 25JAN91 | 119 | / | 78 | X | X | |
| | | 01FEB91 | 109 | / | 80 | | | |
| | | 08FEB91 | 103 | / | 80 | X | X | X |
| | | 15FEB91 | 100 | / | 82 | X | X | |

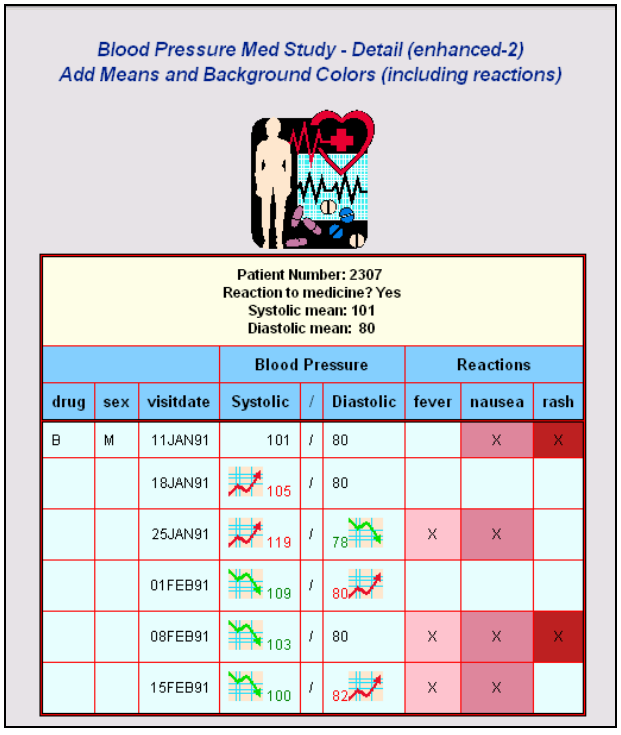
**Figure 3 – Example 3, Partial HTML Output**



Blood Pressure Med Study - Summary (enhanced)
Add Colors

| patient | drug | sex | Reaction? |
|---|---|---|---|
| 1813 | A | F | No |
| 1826 | B | F | Yes |
| 1839 | B | M | No |
| 1852 | B | F | Yes |
| 1865 | B | F | No |
| 1878 | A | F | Yes |
| 1891 | B | M | No |
| 1904 | A | M | No |
| 1917 | B | M | Yes |
| 1930 | A | M | Yes |
| 1943 | B | F | Yes |
| 1956 | B | M | Yes |
| 1969 | B | F | No |
| 1982 | B | M | Yes |
| 1995 | A | M | No |
| 2008 | B | M | Yes |

**Figure 5a – Example 5, Partial HTML Output – Summary Report**

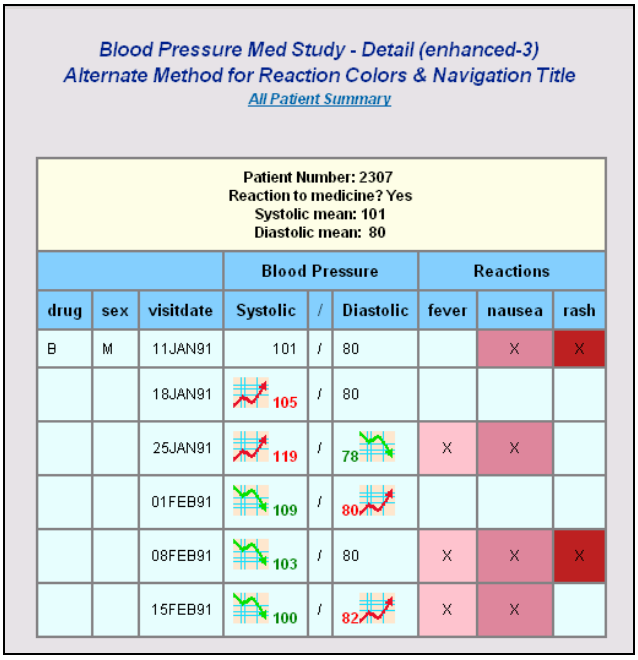


Blood Pressure Med Study - Detail (enhanced-2)
Add Means and Background Colors (including reactions)

Patient Number: 2307
Reaction to medicine? Yes
Systolic mean: 101
Diastolic mean: 80

| | | | Blood Pressure | | | Reactions | | |
|---|---|---|---|---|---|---|---|---|
| drug | sex | visitdate | Systolic | / | Diastolic | fever | nausea | rash |
| B | M | 11JAN91 | 101 | / | 80 | | X | X |
| | | 18JAN91 | 105 | / | 80 | | | |
| | | 25JAN91 | 119 | / | 78 | X | X | |
| | | 01FEB91 | 109 | / | 80 | | | |
| | | 08FEB91 | 103 | / | 80 | X | X | X |
| | | 15FEB91 | 100 | / | 82 | X | X | |

**Figure 4 – Example 4, Partial HTML Output**



Blood Pressure Med Study - Detail (enhanced-3)
Alternate Method for Reaction Colors & Navigation Title
All Patient Summary

Patient Number: 2307
Reaction to medicine? Yes
Systolic mean: 101
Diastolic mean: 80

| | | | Blood Pressure | | | Reactions | | |
|---|---|---|---|---|---|---|---|---|
| drug | sex | visitdate | Systolic | / | Diastolic | fever | nausea | rash |
| B | M | 11JAN91 | 101 | / | 80 | | X | X |
| | | 18JAN91 | 105 | / | 80 | | | |
| | | 25JAN91 | 119 | / | 78 | X | X | |
| | | 01FEB91 | 109 | / | 80 | | | |
| | | 08FEB91 | 103 | / | 80 | X | X | X |
| | | 15FEB91 | 100 | / | 82 | X | X | |

**Figure 5b – Example 5, Partial HTML Output – Detail Report**