

# Advanced Features of PROC REPORT

Kim L. Kolbe Ritzow of Systems Seminar Consultants, Kalamazoo, MI

*Due to the length of this paper output to these examples will not appear in the Proceedings. Copies of the paper in full will be available at the presentation, or can be obtained from the author.*

## Abstract

PROC REPORT is a powerful new Base SAS<sup>®</sup> PROC which combines the functionality of other BASE SAS PROCs such as PRINT, MEANS, FREQ, and TABULATE into a single PROC step.

This paper is designed for users with existing knowledge of PROC REPORT's more basic report writing features and who are interested in learning additional techniques to generate more sophisticated reports.

This paper will focus on customized breaklines, grouped or stacked column headings, and advanced tricks for improving the appearance of reports.

## GROUPing Data in PROC REPORT

Data can be aggregated or grouped in PROC REPORT by defining the variable as a GROUP variable in the DEFINE statement. Also within PROC REPORT we have the ability to control the appearance of the report through the use of options on the PROC and DEFINE statements, such as HEADLINE, HEADSKIP, FORMAT=, WIDTH=, ORDER=, in addition to many other available options. A unique feature of PROC REPORT is the ability to dynamically created newly computed variables through the use of COMPUTE blocks. An example of how some of these statements and options may be used:

```
PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
```

```
RUN;
```

```
PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP;
```

```
COLUMNS REGION STORE DEPT SALEAMT TAX NET;
```

```
DEFINE REGION / GROUP FORMAT=$REGIONF. WIDTH=6
  ORDER=FORMATTED 'Region';
```

```
DEFINE STORE / GROUP ORDER=FORMATTED 'Store';
DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept';
DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2
  WIDTH=10 'Amount/of Sale';
DEFINE TAX / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Sales/Tax';
DEFINE NET / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Net/Sale';
```

```
COMPUTE TAX;
  TAX= SALEAMT.SUM * .05;
ENDCOMP;
```

```
COMPUTE NET;
  NET=SALEAMT.SUM - TAX;
ENDCOMP;
RUN;
```

```
TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';
```

Variables can be grouped based upon a user-defined format. It is a good idea when GROUPing a variable to use the ORDER= option. The default order with PROC REPORT is FORMATTED (based on the formatted value rather than the value itself such as N, E, S, W). THIS IS DIFFERENT FROM OTHER BASE SAS PROCEDURES. All other SAS Procedures use the default ORDER=INTERNAL. PROC REPORT's default may change in future releases to be consistent with other SAS Procedures. Specify ORDER=FORMATTED even though it is the current default. This will assure that PROC REPORT will generate the same report even if the default should change.

Other values that can be specified on the ORDER= option are ORDER=DATA (uses the order of the data coming in), ORDER=INTERNAL (nonformatted value), ORDER=FREQ (order of most popular occurrences), or ORDER=FORMATTED (order of the formatted value, which is current default). All of these options assume ascending order except ORDER=DATA. To reverse the order, use the DESCENDING option in addition to the ORDER= option.

When computing new variables when grouping is in effect, it is important to specify the usage of the variable in the COMPUTE blocks such as SALEAMT.SUM. If the usage of the variable is not

specified the values will appear missing on the report and a message will appear in the SAS Log that the grouping did not take place.

If a report is not grouping properly check to make sure of the following:

- \* Computed variables are defining the statistic usage (SALEAMT.SUM)
- \* There are no variables defined as ORDER variables
- \* GROUPING by enough variables
- \* GROUP variables are left-most columns on report

#### Subtotaling and Grand Totaling Data

Subtotals and grand totals can be created on the report with the use of the BREAK and RBREAK statements:

```
PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
RUN;

PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP;

  COLUMNS REGION STORE DEPT SALEAMT TAX NET;

  DEFINE REGION / GROUP ORDER=FORMATTED
    FORMAT=$REGIONF. WIDTH=6 'Region';
  DEFINE STORE / GROUP ORDER=FORMATTED 'Store';
  DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept';
  DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2
    WIDTH=10 'Amount/of Sale';
  DEFINE TAX / COMPUTED FORMAT=DOLLAR10.2
    WIDTH=10 'Sales/Tax';
  DEFINE NET / COMPUTED FORMAT=DOLLAR10.2
    WIDTH=10 'Net/Sale';

  COMPUTE TAX;
    TAX= SALEAMT.SUM * .05;
  ENDCOMP;

  COMPUTE NET;
    NET=SALEAMT.SUM - TAX;
  ENDCOMP;

  BREAK AFTER REGION/OL SKIP SUMMARIZE SUPPRESS;

  RBREAK AFTER/DOL DUL SUMMARIZE;
RUN;

TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';
```

The options available on the BREAK and RBREAK statements are basically the same, except BREAK supports the SUPPRESS option and RBREAK does not. OL requests a single overline, whereas DOL requests a double overline. If you specify both options on the same statement, the OL option takes precedence. UL requests a single underline, DUL requests a double underline. Similarly, if you specify both options on the same statement, the UL option takes precedence. SKIP will skip one line before the next group begins printing. SUMMARIZE will summarize the data at the group level based on the specified statistic DEFINED (since SALEAMT was defined as an ANALYSIS SUM statistic, SUMMARIZE will SUM the data at the group level). SUPPRESS will suppress the group value from printing on the left-hand side of the report on the break line.

Rather than sub and grand totaling after a group, we could also total before a group. To do so, specify BREAK BEFORE or RBREAK BEFORE. This would print the total before the detail.

Another option available on the BREAK and RBREAK statement is the PAGE option, which will go to a new page after each break.

Make sure as BREAK and RBREAK statements are added that the variable's format widths are increased accordingly to accommodate the size of the totaled value.

**Skipping More Than One Line Between Totals**  
The SKIP option will skip one line before the next group begins printing. To skip more than one line, a COMPUTE block can be used:

```
PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
RUN;

PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP;

  COLUMNS REGION STORE DEPT SALEAMT TAX NET;

  DEFINE REGION / GROUP ORDER=FORMATTED WIDTH=6
    FORMAT=$REGIONF. 'Region';
  DEFINE STORE / GROUP ORDER=FORMATTED 'Store';
  DEFINE DEPT / GROUP 'Dept';
  DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2
```

```

WIDTH=10 'Amount/of Sale';
DEFINE TAX / COMPUTED FORMAT=DOLLAR10.2
WIDTH=10 'Sales/Tax';
DEFINE NET / COMPUTED FORMAT=DOLLAR10.2
WIDTH=10 'Net/Sale';

COMPUTE TAX;
  TAX= SALEAMT.SUM * .05;
ENDCOMP;

COMPUTE NET;
  NET=SALEAMT.SUM - TAX;
ENDCOMP;

BREAK AFTER REGION/OL SKIP SUMMARIZE SUPPRESS;

COMPUTE AFTER REGION;
  LINE '';
  LINE '';
ENDCOMP;

RBREAK AFTER/DOL DUL SUMMARIZE;
RUN;

TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';

```

The LINE statement is specified between in a COMPUTE block between COMPUTE and ENDCOMP statements. The LINE statement can be used to write text in the break lines.

**Changing Group Variable's Value in Summary**  
By default, when using the BREAK statement SAS will print the value of the BREAK variable on the summary line, unless the SUPPRESS option is used, which will prevent this value from printing. A COMPUTE block can be used to print a value other than the BREAK variable's value:

```

DATA SASDATA.STOREDAT;
  LENGTH REGION $14;
  SET SASDATA.STOREDAT;
RUN;

PROC FORMAT;
  VALUE $REGIONF
    'N'      ' = 'NORTH'
    'S'      ' = 'SOUTH'
    'E'      ' = 'EAST'
    'W'      ' = 'WEST'
;
RUN;

PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP;

  COLUMNS REGION STORE DEPT SALEAMT TAX NET;

  DEFINE REGION / GROUP ORDER=FORMATTED
    FORMAT=$REGIONF14. WIDTH=14 'Region';
  DEFINE STORE / GROUP ORDER=FORMATTED 'Store';

```

```

DEFINE DEPT / GROUP 'Dept';
DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2
  WIDTH=10 'Amount/of Sale';
DEFINE TAX / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Sales/Tax';
DEFINE NET / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Net/Sale';

COMPUTE TAX;
  TAX= SALEAMT.SUM * .05;
ENDCOMP;

COMPUTE NET;
  NET=SALEAMT.SUM - TAX;
ENDCOMP;

BREAK AFTER REGION/OL SKIP SUMMARIZE;

COMPUTE AFTER REGION;
  REGION='Regional Total';
ENDCOMP;

RBREAK AFTER/DOL DUL SUMMARIZE;
RUN;

TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';

```

AS LONG AS THE LENGTH OF THE CUSTOMIZED TEXT DOES NOT EXCEED THE LENGTH OF YOUR BREAK VARIABLE A VALUE CAN BE ASSIGNED TO YOUR BREAK VARIABLE IN THE COMPUTE BLOCK. In this example, the length of the BREAK variable (REGION) is a 1-byte character. Several things will need to be done before customized text longer than 1-byte will appear on the report. First, the length of REGION must be set to the length of the customized text. This is done in a Data Step prior to PROC REPORT. Secondly, a length value must be specified on the DEFINE statement's FORMAT= option to prevent truncation after the 5th byte of the customized break (longest formatted value is 5; if it were 7, I would get truncation after the 7th byte, and so on). Finally, the DEFINE statement's WIDTH= option must be adjusted as well to accommodate the width of the column. When the customized break text exceeds the length of the BREAK variable, these three things MUST BE DONE in order for the customized break value to print on the report.

When assigning a value to the BREAK variable in the COMPUTE block, as is being done here, the SUPPRESS option on the BREAK statement has no effect upon whether or not the customized text will appear.

Several LINE statements could have also been specified within this COMPUTE block as was done in an earlier example to print multiple blank lines after or before customized break text.

#### Another Way of Doing It

Rather than going through the various steps as was done in the previous example to adjust the width so the customized break text would print out, the entire printing of the break line could be controlled through a COMPUTE block:

```
PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
RUN;

PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP;

  COLUMNS REGION STORE DEPT SALEAMT TAX NET;

  DEFINE REGION / GROUP ORDER=FORMATTED WIDTH=6
    FORMAT=$REGIONF. 'Region';
  DEFINE STORE / GROUP ORDER=FORMATTED 'Store';
  DEFINE DEPT / GROUP 'Dept';
  DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2
    WIDTH=10 'Amount/of Sale';
  DEFINE TAX / COMPUTED FORMAT=DOLLAR10.2
    WIDTH=10 'Sales/Tax';
  DEFINE NET / COMPUTED FORMAT=DOLLAR10.2
    WIDTH=10 'Net/Sale';

  COMPUTE TAX;
    TAX= SALEAMT.SUM * .05;
  ENDCOMP;

  COMPUTE NET;
    NET= SALEAMT.SUM - TAX;
  ENDCOMP;

  BREAK AFTER REGION/OL;

  COMPUTE AFTER REGION;
    LINE @1 'Regional Totals '
      @37 SALEAMT.SUM DOLLAR10.2
      @49 TAX DOLLAR10.2
      @61 NET DOLLAR10.2;
    LINE '';
  ENDCOMP;

  RBREAK AFTER/DOL DUL SUMMARIZE;
RUN;

TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';
```

When writing the code to control the printing of the break line information notice that we have reverted back to the original code without any adjustments to the user-defined format, FORMAT= and WIDTH= options.

The BREAK statement is controlling the underlining of the values with the OL option and the COMPUTE block is controlling the printing of all the values. Notice on the LINE statement pointers (@) were used to control how and where information will print out. It is important to associate a format with the values on the LINE statement. If formats are not specified, SAS won't display the variable and no error or warning message will appear. When writing out SAS variables notice that the variable's name and the associated statistic (SALEAMT.SUM) were specified. THE STATIC SPECIFIED MUST MATCH THE STATISTIC DEFINED ON THE ANALYSIS STATEMENT FOR THE VARIABLE. ANOTHER STATISTIC CANNOT BE ASSOCIATED WITH THE VARIABLE ON THE COMPUTE BLOCK (such as SALEAMT.MEAN). To associate multiple statistics with the same variable aliases can be used (discussed later in the paper). However, ALIASES CANNOT BE USED IN COMPUTE BLOCKS. The only way to use a variable in a COMPUTE block with a different statistic associated with it other than the one used in the DEFINE statement, a new variable must be created in a Data Step prior to PROC REPORT that contains the same value of the variable you are trying to use.

Notice with computed variables (such as TAX and NET) they are referred to by their computed name and a statistic is NOT associated with them. If a statistic is associated with them (such as TAX.SUM) missing values for that variable will appear on the report. The usage of a computed variable CANNOT be changed to something like TAX.MEAN.

If the SUPPRESS option were specified on the BREAK statement, it would have no effect on the printing of the text in the COMPUTE block.

**Features Supported in the COMPUTE Block**  
The COMPUTE block supports many of the SAS language features such as:

- \* Data Step functions
- \* Macro variables

```

*      Null Data Step statements
*      %INCLUDE statement
*      Select Data Step statements:
*          assignment statements
*          LENGTH statement
*          LINK statement
*          RETURN statement
*          SELECT statement
*          sum statement
*          IF-THEN/ELSE statements
*          GO-TO statements
*          END statement
*          DO (all forms)
*          CALL statements
*      Comments
*      DM statements
*      LINE statement (unique to PROC
      REPORT)

```

### Custom Break Lines

Customized break lines can easily be created by a variation on some of the many things we have just seen. For example:

```

PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
RUN;

PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP;

  COLUMNS REGION STORE DEPT SALEAMT TAX NET;

  DEFINE REGION / GROUP ORDER=FORMATTED WIDTH=6
    FORMAT=$REGIONF. 'Region';
  DEFINE STORE / GROUP ORDER=FORMATTED 'Store';
  DEFINE DEPT / GROUP 'Dept';
  DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2
    WIDTH=10 'Amount/of Sale';
  DEFINE TAX / COMPUTED FORMAT=DOLLAR10.2
    WIDTH=10 'Sales/Tax';
  DEFINE NET / COMPUTED FORMAT=DOLLAR10.2
    WIDTH=10 'Net/Sale';

  COMPUTE TAX;
    TAX= SALEAMT.SUM * .05;
  ENDCOMP;

  COMPUTE NET;
    NET= SALEAMT.SUM - TAX;
  ENDCOMP;

  COMPUTE AFTER REGION;
    LINE '';
    LINE 42* '*';

```

```

LINE '';
LINE '* TOTAL SALES FOR ' REGION $REGIONF. ' WERE:'
      SALEAMT.SUM DOLLAR10.2 '*';
LINE '';
LINE 42* '*';
LINE '';
ENDCOMP;

RBREAK AFTER/DOL DUL SUMMARIZE;
RUN;

TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';

```

Since we are creating customized break text, the BREAK statement has been removed. Now the COMPUTE block is controlling the entire breaking and text writing process.

Just like when PUTting literal text (anything within quotes) we are in control of the entire process, so if blanks within the literal strings are forgotten, text will run together (notice how blanks were left in the literal strings before the value of variables were written out). Similar to the feature available on PUT statement, multipliers (42\* '\*') can be specified which will print the character '\*' forty-two times.

If a location is not specified with a pointer reference (@) SAS will center the text in the COMPUTE block if the CENTER system option is in effect.

If you happen to have a BREAK statement in effect with a SUPPRESS option specified in addition to the COMPUTE block (you shouldn't really need BREAK statement), the values for REGION will not appear on the report. If for some reason a BREAK statement is needed in addition to a COMPUTE block that is being used to write out the values of the break variable DO NOT USE THE SUPPRESS OPTION ON THE BREAK STATEMENT.

Notice on the output generated that the format used when writing out values on the LINE statement determines the amount of space used to write its value. Character variables will fill the space with trailing blanks, whereas numeric variables fill the space with leading blanks.

### Controlling Space and Grand Totals

The trailing blanks on character variables and leading blanks on numeric variables can be controlled by using the \$VARYING. format on the

LINE statement. In addition to using the \$VARYING. format assignment statements are needed to convert values and to check lengths:

```
PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
```

```
;
RUN;
```

```
PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP;
```

```
COLUMNS REGION STORE DEPT SALEAMT TAX NET;
```

```
DEFINE REGION / GROUP ORDER=FORMATTED WIDTH=6
  FORMAT=$REGIONF. 'Region';
DEFINE STORE / GROUP ORDER=FORMATTED 'Store';
DEFINE DEPT / GROUP 'Dept';
DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2
  WIDTH=10 'Amount/of Sale';
DEFINE TAX / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Sales/Tax';
DEFINE NET / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Net/Sale';
```

```
COMPUTE TAX;
  TAX= SALEAMT.SUM * .05;
ENDCOMP;
```

```
COMPUTE NET;
  NET= SALEAMT.SUM - TAX;
ENDCOMP;
```

```
COMPUTE AFTER REGION;
  REGIONC=PUT(REGION,$REGIONF.);
  VARLEN1=LENGTH(REGIONC);
  SALESC =LEFT(PUT(SALEAMT.SUM,DOLLAR10.2));
  VARLEN2=LENGTH(SALESC);
```

```
LINE '';
LINE @17 42* '*';
LINE '';
LINE @17 '* ' TOTAL SALES FOR '
  REGIONC $VARYING. VARLEN1 ' WERE: '
  SALESC $VARYING. VARLEN2 @58 '*';
LINE '';
LINE @17 42* '*';
LINE '';
ENDCOMP;
```

```
COMPUTE AFTER;
  SALESC =LEFT(PUT(SALEAMT.SUM,DOLLAR10.2));
  VARLEN2=LENGTH(SALESC);
```

```
LINE '';
LINE @17 42* '=';
LINE '';
LINE @17 '* ' GRAND TOTAL SALES WERE: '
  SALESC $VARYING. VARLEN2 @58 '*';
LINE '';
```

```
LINE @17 42* '=';
LINE '';
ENDCOMP;
RUN;
```

```
TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';
```

Since the character variable REGION is actually a formatted value, in addition to checking its length with the LENGTH function, the format's value needs to be associated with the variable as well and this is done by using the PUT function. It could have been done all in one step:

```
VARLEN1=LENGTH(PUT(REGION,$REGIONF.));
```

If the character variable was a normal character variable and was not formatted, the PUT would not be necessary, only the LENGTH function would be required. Since the \$VARYING. format is used on character variables, to control the spacing on a numeric variable it must be first converted to a character using the LEFT and PUT functions and then find its length with the LENGTH function. ALL NUMERIC VARIABLES MUST BE CONVERTED TO CHARACTER, THE LENGTH STATEMENT ALONE IS NOT ENOUGH. Similarly this could have all been done in one step:

```
VARLEN2=LENGTH(LEFT(PUT(SALEAMT.SUM,DOLLAR10.2)));
```

Since the text will now vary in length (it will not be padded out with spaces after characters or before numerics), the length of the text string will vary. Therefore, the location of the '\*' literal with pointer references will have to be fixed to a location (like @17) when writing out the text line. Once a pointer reference (@) is used anywhere within the COMPUTE block, the centering option is turned off for the entire COMPUTE block, that is why the location of the '\*' string (@17 42\* '\*') was also fixed. If left as it was in the last example (42\* '\*') the string of asterisks would be left aligned because the centering is automatically turned off.

A grand total has also been added to the report with the COMPUTE AFTER block (note: subtotals are COMPUTE AFTER *variable* blocks). Since the grand totals are customized breaks, the RBREAK statement has been removed.

### Group or Stacked Column Headings

A column can be grouped or stacked by defining it as an ACROSS variable:

```

PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
RUN;

PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP SPLIT='*';

  COLUMNS REGION DEPT SALEDATE;

  DEFINE REGION / GROUP FORMAT=$REGIONF.
    ORDER=FORMATTED WIDTH=6 'Region';
  DEFINE SALEDATE / ACROSS FORMAT=MMDDYY8.
    ORDER=INTERNAL 'Date*of Sale';
  DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept';

  WHERE '07MAY94'D <= SALEDATE <= '08MAY94'D;

RUN;

TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEKEND OF 5/07/94 - 5/08/94';

```

This example subset the dates using a WHERE statement, which is supported on PROC REPORT.

Particularly with date values displayed in one of the three formats that uses slashes to separate values (such as MMDDYY8., DDMMYY8., and YYMMDD8.) it is important to use the SPLIT= option on the PROC statement to change the default split character to something other than a '/'. If it is not changed, the date value will split to a new line each time it sees the '/' in the date value.

By default ACROSS variables will appear based on their formatted values. With dates it is useful to specify the ORDER=INTERNAL option so the dates will appear on the report in date order, not in formatted order.

When a variable is defined as an ACROSS variable and another variable is not specified above or below the ACROSS variable (there are no commas in the COLUMNS statement like in the next example) then the N statistic is displayed by default.

#### Shared Column with an Analysis Variable

An ACROSS variable can share a column with an analysis variable by using a comma between the two variables on the COLUMN statement:

```

PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
RUN;

PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP SPLIT='*';

  COLUMNS REGION DEPT SALEDATE, SALEAMT;

  DEFINE REGION / GROUP FORMAT=$REGIONF.
    ORDER=FORMATTED WIDTH=6 'Region';
  DEFINE SALEDATE / ACROSS FORMAT=MMDDYY8.
    ORDER=INTERNAL 'Date*of Sale';
  DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept';
  DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2;

  WHERE '07MAY94'D <= SALEDATE <= '08MAY94'D;

RUN;

TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEKEND OF 5/07/94 - 5/08/94';

```

The order in which the variables are defined on the column statement will determine how they appear on the report. The first variable specified will display above the second variable specified. A comma separates the two variables, which will create a stacking effect. In this case, the values of SALEDATE will appear above the word SALEAMT. If they were reversed on the column statement the word SALEAMT would appear above the date values on the report, which is not as visually pleasing, but provides the same information.

#### Shared Columns and Multiple Analysis Variables

Computed variables cannot be defined as an across variable, nor can they share a column with an analysis variable. The variable must be previously defined in a Data Step in order for it to be used as an ACROSS variable.

More than one analysis variable can be specified with an across variable on the COLUMNS statement. In this case, it produces a side-by-side comparison of SALEAMT and TAX for the specified SALEDATES.

```

DATA STOREDAT;
  SET SASDATA.STOREDAT;
  TAX=SALEAMT * .05;
RUN;

```

```

PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
RUN;

PROC REPORT DATA=STOREDAT NOWINDOWS HEADLINE
HEADSKIP SPLIT='*';

COLUMNS REGION DEPT (SALEAMT TAX),SALEDATE;

DEFINE REGION / GROUP FORMAT=$REGIONF.
  ORDER=FORMATTED WIDTH=6 'Region';
DEFINE SALEDATE / ACROSS FORMAT=MMDDYY8.
  ORDER=INTERNAL 'Date*of Sale' WIDTH=10;
DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept';
DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2;
DEFINE TAX / ANALYSIS SUM FORMAT=DOLLAR10.2;

WHERE '07MAY94'D <= SALEDATE <= '08MAY94'D;

RUN;
TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEKEND OF 5/07/94 - 5/08/94';

```

Shared columns that have variables of differing widths will use the width of the variable closest to the first row of the report. Use the WIDTH= option to properly set the width, otherwise the values will appear unformatted and increasing the format width will not help.

### Defining a Variable with Multiple Statistics

There are two ways in which a variable can be defined with more than one statistic. One way is to define the associated statistics on the COLUMNS statement:

```

PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
RUN;

PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
HEADLINE HEADSKIP SPLIT='*';

COLUMNS REGION DEPT (MIN MAX MEAN),SALEAMT;

DEFINE REGION / GROUP FORMAT=$REGIONF.
  ORDER=FORMATTED WIDTH=6 'Region';
DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept';
DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2;
RUN;
TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';

```

SALEAMT is defined as an ANALYSIS variable and the statistics specified on the COLUMNS statement are telling SAS to display those statistics above the variable SALEAMT (reversing the order of the statements on the COLUMN statement would place the statistic underneath the word SALEAMT).

A statistic should not be specified on the DEFINE statement. If it is, it has no effect; the statistic specified on the COLUMN statement takes precedence. A usage option is not required because the statistic defined on the COLUMN statement determines the usage.

Valid statistics that can be used are the following (these are the same statistics that can be used on the DEFINE statement for ANALYSIS variables):

|      |        |        |
|------|--------|--------|
| CSS  | NMISS  | SUMWGT |
| CV   | PRT    | T      |
| MAX  | RANGE  | USS    |
| MEAN | STD    | VAR    |
| MIN  | STDERR |        |
| N    | SUM    |        |

### Defining Aliases

Another way of accomplishing the same result as in the previous example is to define aliases:

```

PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
RUN;

PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
HEADLINE HEADSKIP SPLIT='*';

COLUMNS REGION DEPT SALEAMT=MINSALE
SALEAMT=MAXSALE SALEAMT=MEANSALE;

DEFINE REGION / GROUP FORMAT=$REGIONF.
  ORDER=FORMATTED WIDTH=6 'Region';
DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept';
DEFINE MINSALE / ANALYSIS MIN FORMAT=DOLLAR7.2
  'Lowest*Sale';
DEFINE MAXSALE / ANALYSIS MAX FORMAT=DOLLAR7.2
  'Largest*Sale';
DEFINE MEANSALE / ANALYSIS MEAN
  FORMAT=DOLLAR7.2 'Average*Sale';

RUN;
TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';

```

The advantage of defining aliases is there is more control over the appearance of each of the statistics by the use of options on the DEFINE statement.

Aliases cannot be used in COMPUTE blocks nor defined as an ACROSS variable.

### Improving the Appearance of the Report

Within the labels specified on the DEFINE statement, one of six special characters can be specified that will extend the column heading over several columns. This is most useful with variables that are defined as ACROSS variables:

```
DATA STOREDAT;
  SET SASDATA.STOREDAT;
  TAX=SALEAMT * .05;
RUN;
```

```
PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST';
  VALUE DATEFMT
    '07MAY94'D = '07MAY94.'
    '08MAY94'D = '08MAY94.';
RUN;
```

```
PROC REPORT DATA=STOREDAT NOWINDOWS HEADLINE
  HEADSKIP SPLIT='*';
```

```
COLUMNS REGION DEPT SALEDATE,(SALEAMT TAX);
```

```
DEFINE REGION / GROUP FORMAT=$REGIONF.
  ORDER=FORMATTED WIDTH=6 'Region' '-';
DEFINE SALEDATE / ACROSS FORMAT=DATEFMT.
  ORDER=INTERNAL WIDTH=10 ' ';
DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept' '-';
DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2;
DEFINE TAX / ANALYSIS SUM FORMAT=DOLLAR10.2;
```

```
WHERE '07MAY94'D <= SALEDATE <= '08MAY94'D;
```

```
RUN;
TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';
```

The six valid characters that can be used to expand a column heading are: = \_ . \* - +

When you are using a variable's value as a column header (as we have done with the date values), PROC FORMAT honors these six special characters in the first and last positions of the formatted value and will expand them out upon displaying the value.

These special characters can be specified in the label on the DEFINE statement without any text between them, in which case they will underline only the column they are specified on (HEADLINE is a solid line):

```
PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
```

```
;
RUN;
```

```
PROC REPORT DATA=STOREDAT NOWINDOWS
  HEADSKIP SPLIT='*';
```

```
COLUMNS REGION DEPT SALEAMT;
```

```
DEFINE REGION / GROUP FORMAT=$REGIONF.
  ORDER=FORMATTED WIDTH=6 'Region' '-';
DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept' '-';
DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2
  '-';
```

```
RUN;
TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';
```

### Spanning Multiple Columns With a Header

Multiple columns can be spanned with a common column header by specifying the text in the COLUMN statement:

```
PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
```

```
;
RUN;
```

```
PROC REPORT DATA=STOREDAT NOWINDOWS
  HEADSKIP SPLIT='*';
```

```
COLUMNS REGION DEPT ('Weekly Sales' SALEAMT TAX);
```

```
DEFINE REGION / GROUP FORMAT=$REGIONF.
  ORDER=FORMATTED WIDTH=6 'Region' '-';
DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept' '-';
DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2
  '-';
DEFINE TAX / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Sales/Tax' '-';
```

```
COMPUTE TAX;
  TAX= SALEAMT.SUM * .05;
ENDCOMP;
RUN;
TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';
```

Inside parenthesis on the COLUMN statement the spanning text must be specified in single or double quotation marks followed by the list of variables to be spanned.

#### **In Summary**

PROC REPORT is a powerful new Base SAS Procedure which provides us with more control over our report's appearance than any other existing Base SAS Proc.

We have seen in the course of this paper how PROC REPORT can create customized breaklines, create grouped or stacked column headings, and we learned some advanced tricks for improving the appearance of the reports.

While not discussed in this paper, PROC REPORT is capable of doing some things that no other SAS Procedure is capable of like creating multi-column reports and creating percentages on the report.

#### **Trademark Notice**

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

#### **Useful Publications**

SAS Institute Inc. (1990), SAS® Guide to the Report Procedure, Usage and Reference, Version 6, First Edition, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1993), SAS® Technical Report P-258, Using the REPORT Procedure in a Nonwindowing Environment, Release 6.07, Cary, NC.: SAS Institute Inc.

Any questions or comments regarding the paper may be directed to the author:

Kim L. Kolbe Ritzow  
Systems Seminar Consultants  
Kalamazoo Office  
927 Lakeway Drive  
Kalamazoo, MI 49001  
Phone: (616) 345-6636  
Fax: (616) 345-5793