MWSUG 2017 - Paper SA13

Make That Report Look Great with the Versatile Proc Tabulate

Ben Cochran, The Bedford Group, Raleigh, NC

ABSTRACT

Several years ago, one of my clients was in the business of selling reports to hospitals. He used PROC TABULATE to generate part of these reports. He liked the way this procedure 'crunched the numbers', but not the way the final reports looked. He said he would go broke if he had to sell default PROC TABULATE output. So, he wrote his own routine to take TABULATE output and render it through Crystal Reports. This was in during the days of the 6.12 release of SAS® before there was something called the Output Delivery System (ODS). Once he got his hands on SAS ODS, he kissed his Crystal Reports good-bye. This paper is all about using PROC TABULATE to generate fantastic looking reports. If you want to generate BIG money reports with PROC TABULATE, this presentation is for you.

INTRODUCTION

The TABLUATE procedure offers the same statistics as many of the other SAS procedures found in the Base SAS product such as MEANS, SUMMARY and REPORT. In addition to this, the TABULATE procedure provides flexible report writing features such as:

- flexible table construction
- multiple dimensions
- use of labels and formats
- customization with the Output Delivery System (ODS) statements and options.

The general form of the TABULATE step is:

```
PROC TABULATE data = SAS-data-set  options ;
      class  variables ;
      var   variables ;
      table  expression;
run;
```

These are the basic statements to get started with this procedure. The CLASS statement allows you to specify the categorical (or class) variables. The VAR statement allows you to specify analysis variables. The TABLE statement is the real work horse of this procedure. You can use special characters to construct a fairly elaborate report. There are many other statements that can be used to make this procedure quite powerful such as **:**

- **BY**
- **CLASSLEV**
- **FREQ**
- **KEYLEVEL**
- **WEIGHT**

Most of these statements will be illustrated in upcoming examples**.**

The TABLE statement is used to construct the report. The table format along with statistics and variables are specified here. Before variables can be used on the TABLE statement, they must be mentioned on the VAR or CLASS statements first. The 'shape' of the report is controlled by TABLE statement operators.

Operator	Symbol	Task
Comma	**,**	determines the number of dimensions
Asterisk	*****	cross, subgroup or 'within'
Blank		table concatenator
Parenthesis	**( )**	grouping agent
Brackets	**< >**	specifies denominator definitions
Equal	**=**	assigns labels or formats

One of the most important TABLE statement operators to initially focus on is the comma. When there is not a comma present, the report has only one dimension (column). When there is only one comma in the TABLE statement, there are two dimensions (row, column). When there are two commas in the TABLE statement, there are three dimensions (page, row, column).

GETTING STARTED

This paper presents a several examples of PROC TABULATE ranging from fairly simple to more complex that illustrate the power and flexibility of this procedure. The data that is used is derived from the SASHELP.PM dataset. Only the first 18 rows are shown here.

Obs	Hub	Country	TYPE	City	INCOME	OVERHEAD	Year
1	London	Australia	MD11	Acton	288.24	230.59	Last
2	London	Australia	MD11	Acton	523.24	418.59	This
3	London	Australia	MD11	Acton	1,500.24	1,200.19	This
4	London	Australia	MD11	Acton	1,660.57	1,328.45	This
5	London	Australia	DC10	Acton	499.24	394.40	Last
6	London	Australia	DC10	Acton	523.24	413.36	This
7	London	Australia	DC10	Acton	804.24	635.35	This
8	London	Australia	DC10	Acton	874.62	690.95	This
9	San Fran	Australia	A300	Acton	198.24	152.64	Last
10	San Fran	Australia	A300	Acton	523.24	402.89	This
11	San Fran	Australia	A300	Acton	1,308.24	1,007.34	This
12	San Fran	Australia	A300	Acton	1,340.82	1,032.43	This
13	New York	Australia	MD11	Melbourne	529.00	322.69	Last
14	New York	Australia	MD11	Melbourne	1,170.00	713.70	This
15	New York	Australia	MD11	Melbourne	1,596.00	973.56	This
16	New York	Australia	MD11	Melbourne	1,876.61	1,144.73	This
17	Sydney	Australia	DC10	Melbourne	628.71	421.24	Last
18	Sydney	Australia	DC10	Melbourne	251.29	168.36	This

Figure 1. .

EXAMPLE 1:

Use multiple TABLE statements in one step to generate multiple reports.

.

```
PROC TABULATE data = SASHELP.pm;
      class  hub ;
      var    income;
      table  hub * n;
      table  hub * income * sum ;
run;
```

Example 1 program.

Hub					
Frankfrt	London	New York	San Fran	Sydney	Tokyo
N	N	N	N	N	N
92.00	76.00	96.00	76.00	112.00	60.00

Hub					
Frankfrt	London	New York	San Fran	Sydney	Tokyo
INCOME	INCOME	INCOME	INCOME	INCOME	INCOME
Sum	Sum	Sum	Sum	Sum	Sum
99128.02	161560.84	73164.91	102212.65	238586.14	133330.27

Example 1 Output.

Because there are NO commas in either TABLE statement, both reports are only one dimensional (column). There is a column for each value of HUB. The first TABLE statement (Table HUB * N ;) generates a report that shows a frequency count (N statistic) per HUB. The default format, 12.2, controls the number of decimal places. The second TABLE statement (Table HUB * INCOME * SUM ;) generates a report that has a column for each HUB and displays the total INCOME for that HUB.

EXAMPLE 2:
Use multiple CLASS variables to generate a two dimensional report.

```
PROC TABULATE data = SASHELP.pm format=comma12.2;
      class  year  type ;
      var    income;
      where type in ('777', '747');
      table year, type * income * sum ;
run;
```

Example 2 program.

	TYPE	
	747	777
	INCOME	INCOME
	Sum	Sum
Year		
Last	23,999.19	45,720.71
This	131,694.67	177,846.53

Example 2 output

The FORMAT = option on the PROC statement controls the appearance of the whole report. On the TABLE statement, there is one comma, which means the report has 2 dimensions. Starting with the keyword TABLE and moving to the left, everything up to the comma goes in the **row** dimension. Everything after the comma goes in the column dimension. You can have a third, or page, dimension in your report, but this paper only shows one and two dimensional reports.

EXAMPLE 3:
Display multiple statistics and use an OUT = option to create a SAS dataset,

```
PROC TABULATE data = SASHELP.pm format=comma12.2 out = pm;
      class  hub ;
      var    income;
      table  hub, income * (n sum mean max mode);
run;
```

Example 3 program.

This step generates two forms of output. The first is the report that is generated from the procedure, and the second is the output dataset created from the OUT= option on the PROC statement. Both types of output are shown in this paper.

VIEWTABLE: Work.Pm

	Hub	_TYPE_	_PAGE_	_TABLE_	INCOME_N	INCOME_Sum	INCOME_Mean	INCOME_Max	INCOME_Mode
1	Frankfrt	1	1	1	92	99128.018	1077.4785	4703.85668	435.15
2	London	1	1	1	76	161560.84	2125.8005	19833.9489	523.235294
3	New York	1	1	1	96	73164.905	762.13443	3615.84581	556
4	San Fran	1	1	1	76	102212.65	1344.9032	7100.00284	.
5	Sydney	1	1	1	112	238586.14	2130.2334	14595.2537	428.4
6	Tokyo	1	1	1	60	133330.27	2222.1711	17300.7989	.

Example 3 output – the WORK.PM dataset

	INCOME				
	N	Sum	Mean	Max	Mode
Hub					
Frankfrt	92.00	99,128.02	1,077.48	4,703.86	435.15
London	76.00	161,560.84	2,125.80	19,833.95	523.24
New York	96.00	73,164.91	762.13	3,615.85	556.00
San Fran	76.00	102,212.65	1,344.90	7,100.00	.
Sydney	112.00	238,586.14	2,130.23	14,595.25	428.40
Tokyo	60.00	133,330.27	2,222.17	17,300.80	.

Example 3 output – the report.

EXAMPLE 4:
Use an **'='** operator to create blank labels for TYPE and SUM.

```
PROC TABULATE data = SASHELP.pm format=comma12.2 ;
      class  hub  type;
      var    income;
      where type in('777', '747');
      table  hub, type = ' ' * income * sum = ' ';
run;
```

Example 4 program.

The **'='** can be used to display text in the report. On the TABLE statement, it is used twice: once after the TYPE variable, and once after the SUM statistic. Both times, the text is set to a blank. This has the effect of 'blanking' out the words TYPE and SUM. Notice that they are in Example 2 output. TYPE is at the top of the report in line one, and SUM is on line 3 of that report. In this example, those words are not displayed. .

	747	777
	INCOME	INCOME
Hub		
Frankfrt	40,993.42	6,290.07
London	14,024.67	26,122.65
New York	14,547.88	15,100.79
San Fran	24,174.57	32,769.05
Sydney	56,210.43	63,354.74
Tokyo	5,742.90	79,929.94

Example 4 output.

EXAMPLE 5:

Enhance the program by adding the **'f='** and the **rts** option**.** Include the special **ALL** variable**.**

```
PROC TABULATE data = SASHELP.pm format=comma12.2 ;
      class  hub  type;
      var    income;
      where type in('777', '747');
      table  hub all, type * income * sum = ' '
             all = 'Total' * income * sum = ' ' f=dollar12.2 / rts = 12;
run;
```

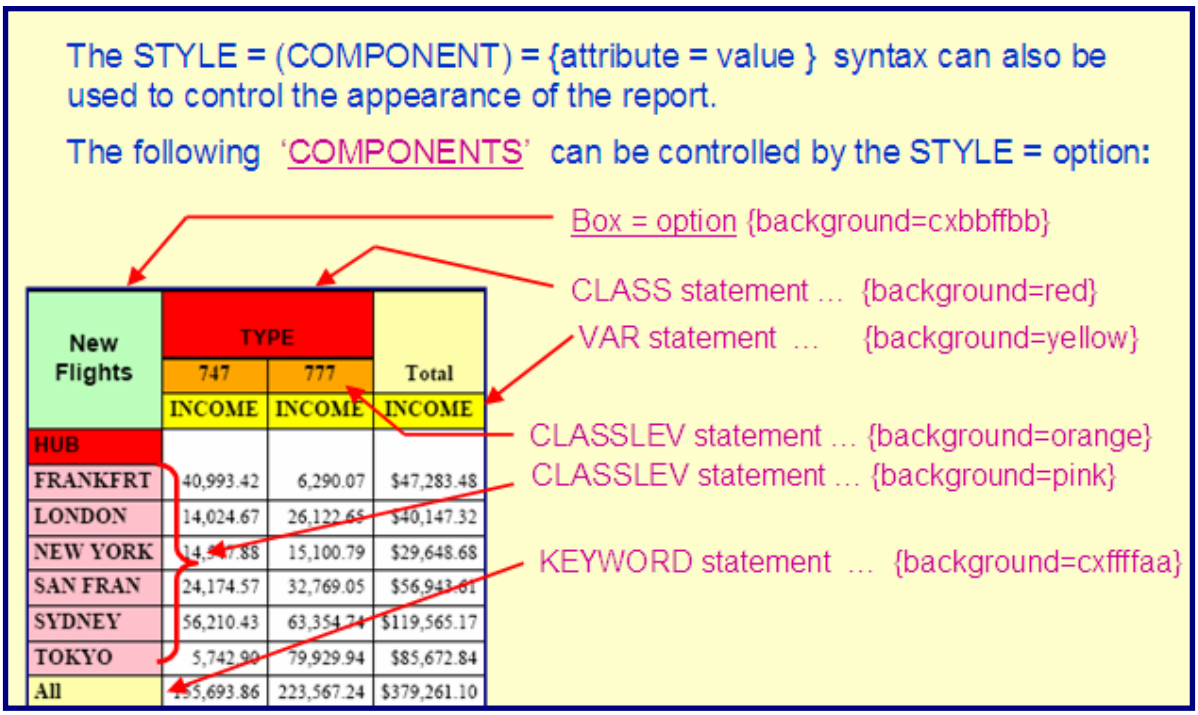
Example 5 program

	TYPE		
	747	777	Total
	INCOME	INCOME	INCOME
HUB			
FRANKFRT	40,993.42	6,290.07	$47,283.48
LONDON	14,024.67	26,122.65	$40,147.32
NEW YORK	14,547.88	15,100.79	$29,648.68
SAN FRAN	24,174.57	32,769.05	$56,943.61
SYDNEY	56,210.43	63,354.74	$119,565.17
TOKYO	5,742.90	79,929.94	$85,672.84
All	155,693.86	223,567.24	$379,261.10

Example 5 output.

Column and row totals were added to this example by using the special variable ALL in both dimensions. ALL creates totals in either the row or the column dimension. In the column dimension, the **='Total'** option was used so that ALL is not displayed like it is in the row dimension. Also, the **f=** option is used in the column dimension to control the format for the total column. Notice the ALL column is the only column where dollar signs appear.

Output Delivery System (ODS) components can be used to enhance this report. Although there are many components to ODS, this paper only looks at the **STYLE=** option. The screen capture below illustrates the type of things that can be done to add color to the last report.



This chart serves as the 'gameplan' for the next report.

EXAMPLE 6:
Use the **STYLE = options** in the PROC TABULATE step to produce a 'colorful' report.

```
ods pdf file='mypdf.pdf';

proc tabulate data=sas_1.pm format = comma12.2 ;
     class hub type / style={font_face=arial background=red};
     classlev hub   / style={background=pink};
     classlev type  / style={background=orange};
     var income     / style={background=yellow};
     keyword all    / style={background=cxffffaa};
     where type in('747','777');
     table hub all, type * income *sum=' '
           all='Total' * income * sum=' '*f=dollar12.2
           / rts=12 box={label='New Flights'
             style={background=cxbbffbb  font_face=arial font_size=4}};
run;

ods pdf close;
```

Example 6 program.

New Flights	TYPE		Total
	747	777	
	INCOME	INCOME	INCOME
HUB			
FRANKFRT	40,993.42	6,290.07	$47,283.48
LONDON	14,024.67	26,122.65	$40,147.32
NEW YORK	14,547.88	15,100.79	$29,648.68
SAN FRAN	24,174.57	32,769.05	$56,943.61
SYDNEY	56,210.43	63,354.74	$119,565.17
TOKYO	5,742.90	79,929.94	$85,672.84
All	155,693.86	223,567.24	$379,261.10

Example 6 output.

EXAMPLE 7:

Modify the program by using different style attributes. Define a URL to be used as a hyperlink in the BOX= option. Link the PROC TABULATE output to a spreadsheet. Note: The ODS statements directing this output to a PDF are not included in this screen capture.

```
proc tabulate data=sas_1.pm f=15.2 s={font_face=arial } ;
     var income;
     class hub type;
     where type in('747', '777');
     keyword all;
     table hub all, type * income * sum=' '
           all='Total' * income * sum = ' ' * f=dollar12.2 /
           rts=12 box={label='New Flights' style=
           { url='c:\newflights.xls' background=light yellow} } ;
run;
```

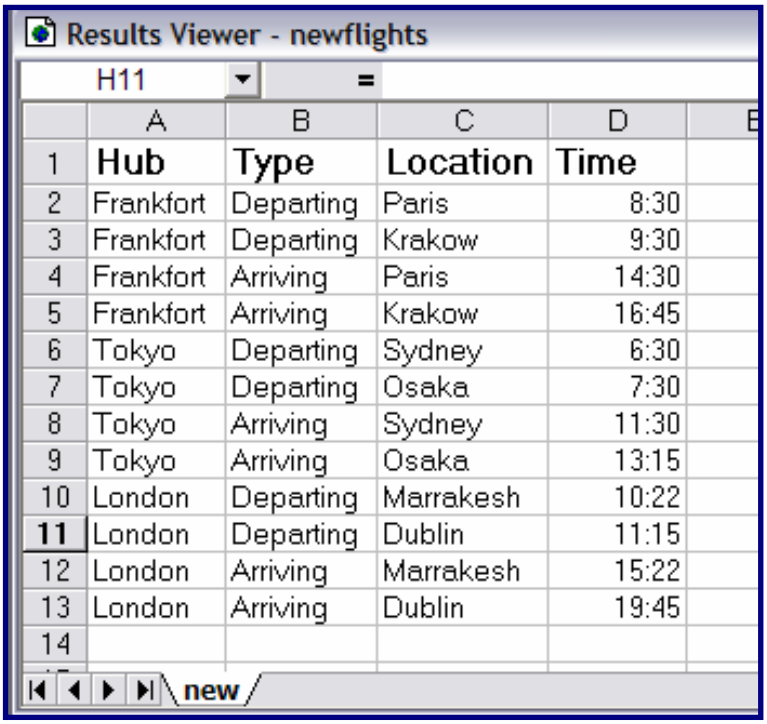
Example 7 program.

Notice the **URL=** option within the BOX=option. This creates a link for the label 'New Flights'. When this label is selected in the report, the newflights.xls spreadsheet opens.

New Flights	TYPE		Total
	747	777	
	INCOME	INCOME	INCOME
HUB			
FRANKFRT	40993.42	6290.07	$47,283.48
LONDON	14024.67	26122.65	$40,147.32
NEW YORK	14547.88	15100.79	$29,648.68
SAN FRAN	24174.57	32769.05	$56,943.61
SYDNEY	56210.43	63354.74	$119,565.17
TOKYO	5742.90	79929.94	$85,672.84
All	155693.86	223567.24	$379,261.10

Example 7 output.

The text 'New Flights' has been defined as the link to the spreadsheet. When you click on 'New Flights', the spreadsheet below opens.



NewFlights.xls.

DOING MORE WITH STATISTICS

The TABULATE procedure can calculate the following descriptive statistics:

- **COLPCTN COLPCTSUM NMISS MIN MAX VAR CV MODE**
- **KURTOSIS ROWPCTN ROWPCTSUM SUMWGT CSS USS RANGE STD**
- **SKEWNESS REPPCTN REPPCTSUM STDERR SUM MEAN STDERR N**
- **LCLM UCLM PAGEPCTN PAGEPCTSUM PCTN PCTSUM STD STDDEV**

Starting with example 8, the SASHELP.CLASS dataset is used to generate PROC TABULATE reports.

EXAMPLE 8:
Illustrate the **N, SUM, PCTN** and the **PCTSUM** statistics. Note: the following program was run in Enterprise Guide. The STYLE = option on the TABLE statement gives the TOTAL row a yellow background.

```
proc tabulate data = sashelp.class format=comma12.2;
    class sex age;
    var   height ;
    table age all='Total' * ( [style = [ background=yellow] ] ),
          height * (n sum pctn pctsum);
    keyword all / style = {background=yellow};
run;
```

Example 8 program

	Height			
	N	Sum	PctN	PctSum
Age				
11	2	108.80	10.53	9.19
12	5	297.20	26.32	25.09
13	3	184.30	15.79	15.56
14	4	259.60	21.05	21.92
15	4	262.50	21.05	22.16
16	1	72.00	5.26	6.08
Total	19	1,184.40	100.00	100.00

Example 8 output.

PCTN and **PCTSUM** are calculated as follows:
The TOTAL number of observations (**N**) is 19. For each cell in the PCTN column the formula is:
PCTN (cell) = N / 19 * 100.

The **SUM** of all the student's HEIGHT is 1,184.40.
PCTSUM (cell) = SUM / 1184 * 100.

EXAMPLE 8B:
Illustrate the N, SUM, **ROWPCTN** and the **ROWPCTSUM** statistics. A second class variable (SEX) is added to the TABLE statement to give the report a two dimensional appearance. Because the new class variable is in the column dimension, there will be a group of columns for each gender.

```
proc tabulate data = sashelp.class format=comma12.2;
     class sex age;
     var   height ;
     table age, sex * height * (n sum rowpctn rowpctsum)
 run;
```

Example 8B program.

	Sex							
	F				M			
	Height				Height			
	N	Sum	RowPctN	RowPctSum	N	Sum	RowPctN	RowPctSum
Age								
11	1	51.30	50.00	47.15	1	57.50	50.00	52.85
12	2	116.10	40.00	39.06	3	181.10	60.00	60.94
13	2	121.80	66.67	66.09	1	62.50	33.33	33.91
14	2	127.10	50.00	48.96	2	132.50	50.00	51.04
15	2	129.00	50.00	49.14	2	133.50	50.00	50.86
16	.	.	.	.	1	72.00	100.00	100.00

Example 8B output.

When the two **RowPctN** columns are added together, they total 100. Also, when the two RowPctSum columns are added together, they also total 100. For **RowPct** statistics, the formula is as follows:

RowPctN = N (single cell) / Row Total for N.

RowPctSum = SUM (single cell) / Row total for Sum.

EXAMPLE 8C:

Use the denominator definitions as an alternative way to create percentages that add to 100 across the row. Prior to the development of RowPctN and RowPctSum, denominator definitions ' < > ' were used to control the calculation of the statistics.

```
proc format;
     picture pfmt low-hight = '   009.99%';
proc tabulate data = sashelp.class format=comma12.2;
    class sex age;
    var   height ;
    table age, sex * height * (n sum pctn<sex> pctsum<sex>*f=pfmt.);
run;
```

Example 8C program.

Notice the use of Proc FORMAT to create a Percent 'picture'. It is applied only to the **PctSum** column.

	Sex							
	F				M			
	Height				Height			
	N	Sum	PctN	PctSum	N	Sum	PctN	PctSum
Age								
11	1	51.30	50.00	47.15%	1	57.50	50.00	52.84%
12	2	116.10	40.00	39.06%	3	181.10	60.00	60.93%
13	2	121.80	66.67	66.08%	1	62.50	33.33	33.91%
14	2	127.10	50.00	48.95%	2	132.50	50.00	51.04%
15	2	129.00	50.00	49.14%	2	133.50	50.00	50.85%
16	.	.	.	.	1	72.00	100.00	100.00%

Example 8C output.

EXAMPLE 8D:

Modify the previous program by adding **STYLE =** options and the Keyword ALL to the Column dimension. Note that **STYLE =** can be abbreviated as **S=.**

```
proc tabulate data = sashelp.class format=comma12.2;
     class sex age;
     var   height ;
     table age, sex * height * (n sum rowpctn*[s=[background=pink] ]
                                rowpctsum*[style=[background=yellow] ] )
                all * height * (n sum rowpctn='Percent'*[s=[background=orange] ] );
     keyword rowpctn   / style=[background=red];
 run;
```

Example 8D program.

Also notice that since both **RowPctN** and **RowPctSum** add up to 100, only one statistical column is used in the ALL section of the column dimension.

	Sex								All		
	F				M						
	Height				Height				Height		
	N	Sum	RowPctN	RowPctSum	N	Sum	RowPctN	RowPctSum	N	Sum	Percent
Age											
11	1	51.30	50.00	47.15	1	57.50	50.00	52.85	2	108.80	100.00
12	2	116.10	40.00	39.06	3	181.10	60.00	60.94	5	297.20	100.00
13	2	121.80	66.67	66.09	1	62.50	33.33	33.91	3	184.30	100.00
14	2	127.10	50.00	48.96	2	132.50	50.00	51.04	4	259.60	100.00
15	2	129.00	50.00	49.14	2	133.50	50.00	50.86	4	262.50	100.00
16	.	.	.	.	1	72.00	100.00	100.00	1	72.00	100.00

Example 8D output.

EXAMPLE 9:

Write a PROC TABULATE step that illustrates the use of the **ColPctN** and **ColPctSum** statistics.

```
proc tabulate data = sashelp.class format=comma12.2;
     class sex age;
     var   height ;
     table age all, sex * height * (n sum colpctn colpctsum);
run;
```
Example 9 program.

Notice that the keyword ALL has been added to the ROW dimension. The **ColPctN** and **ColPctSum** columns add up to 100 percent within each value of the SEX variable.

	Sex							
	F				M			
	Height				Height			
	N	Sum	ColPctN	ColPctSum	N	Sum	ColPctN	ColPctSum
Age								
11	1	51.30	11.11	9.41	1	57.50	10.00	9.00
12	2	116.10	22.22	21.29	3	181.10	30.00	28.34
13	2	121.80	22.22	22.34	1	62.50	10.00	9.78
14	2	127.10	22.22	23.31	2	132.50	20.00	20.73
15	2	129.00	22.22	23.66	2	133.50	20.00	20.89
16	.	.	.	.	1	72.00	10.00	11.27
All	9	545.30	100.00	100.00	10	639.10	100.00	100.00

Example 9 output.

CONCLUSION

TABULATE is a very useful procedure in the SAS report writing arsenal. It can deploy many statistics to crunch the numbers, then display them in a number of ways. The addition of ODS as an enhancement with version 8 SAS makes this procedure even more valuable.

ACKNOWLEDGMENTS

As always, I would like to thank the people in the Technical Support department of SAS Institute for their kind and helpful assistance for this past year. Their knowledge helped make this presentation possible.

CONTACT INFORMATION

If you have any questions or comments, the author can be reached at:

Ben Cochran
The Bedford Group
3224 Bedford Avenue
Raleigh, NC 27607
Work Phone: 919.741.0370
Email: bencochran@nc.rr.com