

## The Art of Being Color-Communication-Effective, Not Just Colorful

LeRoy Bessler PhD, Bessler Consulting and Research  
Fox Point, Milwaukee, Wisconsin, USA, Le\_Roy\_Bessler@wi.rr.com

### Abstract

Though it includes some information and coding specific to SAS, SAS/GRAPH, and ODS, this tutorial is also a software-independent guide to using color to communicate, rather than decorate. Color does more than merely add visual excitement to your output. When it comes to the pitfalls of, and best practices for, using color, you don't know what you don't know. Come and find out.

### Introduction

Though it includes information and coding specific to SAS®, SAS/GRAPH®, and ODS, this paper is also a software-independent guide to using color to communicate, rather than decorate. Color does more than merely add visual excitement to your output. Some visual communication situations require color—the human eye can reliably distinguish only five shades of gray (actually, five shades of *any* one color). The commonest form of color blindness cannot distinguish red from green—yet red, yellow, and green is (unnecessarily) the most popular form of color-coding for information, the so-called “Traffic Lighting”. Furthermore, it is not uncommon to encounter nearly unreadable black text on a medium-to-dark blue background, or even yellow on white. There is a best color system for web publishing, and a special small subset (now of diminishing practical importance) of that 16.7-million-color palette. And there is a convenient-to-use color system for print-only publishing. Color swatches can be generated with just a few SAS statements, and sample charts for evaluating text-background color combinations are not much more difficult. For high-volume samples, macro-based color design tools are also provided. This paper was created with SAS 8.2, but is still relevant for Version 9.2 of SAS. The intended audience is all levels of SAS users, and users of any other software that creates color output.

This paper is based on decades of my working with color for communication, starting in the days when color devices for computer output were uncommon, were not cheap, and did not deliver very good results. Besides being a user of color technology, I was involved in evaluation, selection, and deployment of color displays, color printers and plotters, and color copiers (including the first high-resolution, feature-and-function-rich digital copier). My interest in communication with color is really just one aspect of a wide and deep interest in visual communication of computer-sourced information.

I regret that I cannot provide bibliographic citations for the research study results cited immediately below, nor for the remarks attributed later to some experts. Such information is drawn from notes taken from reading and listening many years ago. Work by these people, and more recent research reports on color, can be found via web search. For a web search, also use the British spelling “colour”.

Among the benefits of color reported in various studies are: (1) increased readership; (2) increased reading speed and comprehension; (3) faster learning; (4) reduced error rates; and (5) improved recognition, recall, and response.

This paper is an updated and enhanced republication of my second major work on color, Reference 2.

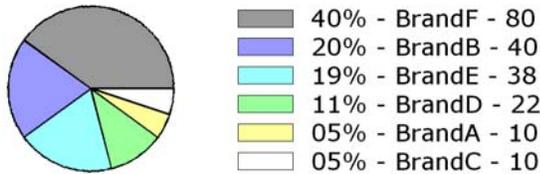
# Using Color for Communication

## Color Does Not Improve Bad Design

### Use Color To Communicate, Not To Decorate

#### Pie Chart With Legend Communicates with Color

Market Share, Brand, and Sales in Billions of Units

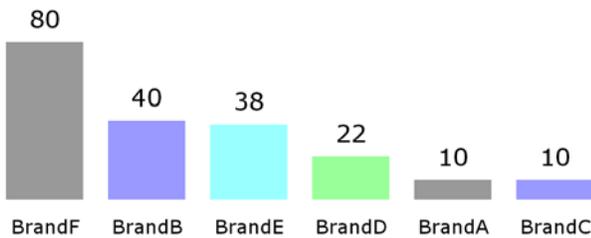


This pie chart uses color to communicate. If your visual communication has no need to distinguish response levels or categories, use Black and White, or some other color pair for foreground and background. If you have a few levels or categories, gray shades may suffice. If you have many levels or categories, color is necessary. It is impossible to reliably distinguish more than five shades of a single hue. Of course, you may be able to expand your palette with Black and/or White, depending on the application and the background color.

## Use of Color Can Confuse, Rather Than Communicate

### Confusing Color Use in a Business Magazine

Units Sold (in Billions) by Brand



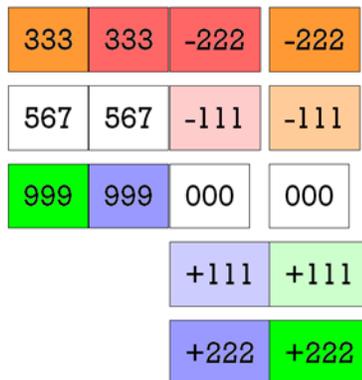
Viewers attribute significance/meaning to your use of color, even when none is intended. So, be careful what you do, whenever you use color. Use of color without a design objective can disorient, confuse, or even mislead the viewer. Failed person-to-person communication is always the fault of the transmitter, not the receiver. The content of the example at the left is different from the magazine illustration I saw, but the misuse of color is exactly parallel. There is NO relationship between BrandF and BrandA, and none between BrandB and BrandC.

## For Those Who Can't See a Color Difference, There Is None

ODS or Widget Traffic Lighting



Instead, Author Recommends "Flag Lighting" Alternatives\*



*The commonest color blindness cannot distinguish red and green*, a frequently used color combination in the USA. Prof. Jay Neitz of the Eye Institute of the Medical College of Wisconsin reported that 8 to 10 percent of American males have some form of color blindness, but, due to genetic differences, only about one-half percent of American females. (I cannot provide the bibliographic citation for this information. I read it in a local Milwaukee newspaper several years ago. I expect that you can verify it via a web search.)

\*Figure 1 in Appendix A uses "Irish Flag Lighting", but with gray substituted for white.

## Maximize Color Contrast between Text and Background

Contrast between foreground and background is essential to communication. ODS opened the door to “enhancing” tables with color. Besides the unfortunately popular Traffic Lighting, there are problems using Yellow with White, or Black (or other dark) text on dark or intense background colors. Evaluate the text-background combinations in the illustrations below. See also the contrast demonstration charts in Figures 6 & 7. It should be noted that adequate contrast for online display does not guarantee the same for hardcopy, which is not brightly backlit.



## Make Colored Text and Lines Thicker, Colored Symbols Bigger

Use of Black and White for print in newspapers, magazines, and books is no accident. Their high contrast makes them the most readable foreground-background combination. Colored text and lines are harder to see. Colored lines should be thickened. SAS/GRAPH enables this with the W= option for plot lines in the SYMBOL statement, with the SHAPE=LINE(width-number) option in the LEGEND statement, and with the WOUTLINE= option for the statements used with the GCHART and GMAP PROCs. ODS lets you specify Bold for fonts, and SAS/GRAPH has bold versions of many of its own software fonts, as well as allowing you to use Windows TrueType fonts with Bold (e.g., as in f= 'Georgia/Bold').

Colored symbols also need to be bigger. Use h= on the SYMBOL statement.

## Jan White on Color Communication

- If everybody screams, all you get is noise. The less color is used, the more effective it is.
- Color consistency provides recognition.
- Use color to sort and/or link information.
- Make large areas pale, small areas bright.
- Don't waste color on titles—for emphasis, use large or bold print instead.

## Michael Turton on Color Communication

- Color works better with space around it.
- Color prioritizes information, whether it is meant to or not.

## Aaron Marcus on Color Communication

- Use blue for large areas, not text or lines. Blue-sensitive color receptors are the least numerous in the retina's central focusing area.
- Use red or green in the center of the visual field. The edges of the retina are not very sensitive to these colors.

## The Case for “UnColor”, and How To Use It

### When to Use or Not Use Color on a Graph

- If you have no response levels/categories, use black & white.
- For a few levels or categories, gray shades may suffice.
- For many levels or categories, color is necessary.

### Benefits of Boring Black-and-White

Technology to print black and shades of gray is faster, cheaper, and more reliable.

Black, white, and shades of gray are easier to use. Not only is the equipment simpler, but also their use requires no agonizing over color selection.

Finally, such output is more copyable. Regardless of the proliferation of cheap color printers at work and at home, the copiers that you find in abundance in the workplace are still almost always black-and-white. Why does that matter? Well, good graphs, maps, and tables—if hardcopy—will get copied when people want to share them.

### SAS/GRAPH Names for Grays

Light Gray, Medium Gray, and Dark Gray (e.g., CXCCCCCC, CX999999, and CX666666), even with White and Black, may not provide enough colors. If so, use color names of the form GRAYll, where ll is a hexadecimal code with range 00-FF. FF (hex for decimal 255) is 0% gray, i.e., WHITE. 00 (hex for decimal 0) is 100% gray, i.e., BLACK. 80 (hex for decimal 128) is 50% gray.

Here are other correspondences for your possible use:

D5: 17%, CC: 20%, C0: 25%, AA: 33%, 99: 40%, 66: 60%, 55: 67%, 40: 75%, 33: 80%, 2B: 83%

Unfortunately, however, the very dark shades of gray tend to be unusable.

### How to Choose/Use SAS/GRAPH Grays

Gray shades too close together are difficult or impossible to distinguish.

Here is a theoretical algorithm for creating a gray color palette. Decide how many grays, N, are needed for the chart, divide 256 by N - 1, and use the quotient (in hexadecimal) as the increment from 00 to FF for ll in GRAYll assignments. Subsets of the values provided in the section above can produce equally spaced grays for sets of 3, 4, 5, 6, or 7 PATTERN statements.

As noted earlier, the human eye cannot reliably distinguish more than five shades of gray (or of any other color), and dark grays are problematic. Hence, I characterize this algorithm as “theoretical”.

Sometimes gray shades do not photocopy well. And black text on a gray background can be a problem.

**NOTE: Grays with names of the form GRAYll are not browser-safe. See the next section.**

## For the Web, It Once Was More Important to Use “Browser-Safe” or “Web-Safe” Colors

### When and Why?

At one time, many web users still had displays or video cards limited to 256 colors. Even when the display and the video card had a higher capability, the video card might have been set to display only 256 colors. When I was working as a consultant prior to May 2003, I did encounter sites where PC’s were limited to 256 colors. Today, your users of older technology mobile devices might be limited to 256 colors. **The rest of this section takes up this now diminishing problem and its circumvention.**

To check or change the setting of your video card on a Windows computer, click Start > Settings > Control Panel > Display > Settings > Colors.

To deal with equipment diversity, web browsers determine the currently set limits of the display unit’s video card, and, if needed, will remap unsupported colors. (Compare Figures 8 and 9.)

Video displays produce colors as combinations of Red, Green, and Blue, the RGB color system. All web browsers agree on a universal common subset of 216 browser-safe RGB colors.

They are RGB colors with names, in SAS, of the form CXrrggbb. The web-safe RGB colors restrict rr, gg, and bb to the six values 00, 33, 66, 99, CC, FF, which correspond to 0%, 20%, 40%, 60%, 80%, 100% of red, green, and blue. (216 = 6 X 6 X 6.)

If a web browser detects a color outside this set on a web page to be shown on a 256-color display, it remaps the color to a browser-safe one. Then, Web Designer Color does not equal Web Viewer Color. There are 16,777,216 RGB colors, but only 216 are browser-safe.

All of the SAS predefined color names (see below) and all of the HTML color names (see below) have RGB equivalents, but only seven of each are browser-safe.

SAS GREEN, contrary to the RGB value still listed in the Version 8 manual, was changed in Version 6.12, and is no longer browser-safe—even though Green is one of the three RGB primaries. The new SAS name, and the HTML name, for browser-safe green is “LIME”. I agree that browser-safe green is perhaps not exactly what most people consider to be a “typical” green (but “typical” being vague, imprecise, and inherently subjective). Browser-safe color CX009900 can serve well as a typical green.

See Figure 2 in Appendix A for 81 samples of browser-safe colors. The basic colors are Red (CXFF0000), Yellow (CXFFFF00), Green (CX00FF00), Cyan or Turquoise (CX00FFFF), Blue (CX0000FF), Magenta (CXFF00FF), Black (CX000000), and White (CXFFFFFF). The upper chart shows the only way for RGB colors to vary in lightness with constant hue.

If you study the full set of 216 browser-safe colors in Figure 5, you may conclude, as I have, that from the browser-safe palette it is difficult to select subsets of “related” colors, other than those in Figure 2. For how to add gray to each of the browser-safe primaries and secondaries, see Figure 3. Another selection of small sets of related browser-safe colors is presented in Figure 4.

NOTE: This seeming limitation can actually be a benefit. A palette of “only” 216 colors does reduce the opportunity for needless agonizing about which colors to use. Presentation of computer-sourced information or charts does not have the same palette requirements as painting a portrait or a landscape.
---

## How To See the Effect of Browser-Unsafe Colors (Compare Figures 8 and 9)

You need a display unit and video card that can display more than 256 colors. As explained in the prior section, use the Control Panel to verify that your video card is set to display more than 256 colors.

Either by using the code in Appendix B, or with any other means that you like, create a web page that includes easily visible patches of SAS Predefined Colors BLUE, TAN, and CREAM. BLUE is one of the only seven web-safe SAS Predefined Colors.

Open the web page with your web browser. The colors will look OK. Close your web browser.

Use the Windows Control Panel to change your video card to display only 256 colors.

Now re-open the web page with your web browser. It will detect the video card's color impoverishment. You will see that the browser has remapped TAN and CREAM, with the browser-safe color subset.

**Be sure to reset your video card back to its normal setting.**

## The Best Color System for Doing Hardcopy Only

It is easy to vary lightness with constant hue by using the HLS color system. When your target is hardcopy, HLS colors are an excellent choice, also providing easy tunability of transition in hue and "saturation". HLS color names are of the form *Hhhllss*. Here is how they work:

- *hhh* is the hexadecimal code for Hue
- *ll* is the hexadecimal code for Lightness (also called "Luminance")
- *ss* is the hexadecimal code for Saturation
- *hhh, ll, ss* ranges are 000-168, 00-FF, 00-FF
- *hhh* = 000 - 168 defines a "wheel of hues", 0 - 360 degrees
- *ll* = 00 (0%) always produces black, regardless of hue or saturation
- *ll* = FF (100%) always produces white, regardless of hue or saturation
- *ss* = FF (100%) always produces the fully saturated hue
- *ss* = 00 (0%) always produces a gray, regardless of selected hue
- *llss* = 80FF is what I call the "true color"

There are six/seven special hues (primary colors and their combinations) in the HLS color wheel.

<b>hhh</b>	<b>color</b>	<b>position</b>
000	Blue	0 degrees
03C	Magenta	60 degrees
078	Red	120 degrees
0B4	Yellow	180 degrees
0F0	Green	240 degrees
12C	Cyan (Turquoise)	300 degrees
168	Blue	360 degrees

In this scheme, Violet lies between Blue and Magenta, Orange between Red and Yellow, Yellow-Green between Yellow and Green, etc. To get to these other colors, and to adjust their precise hue, you have to

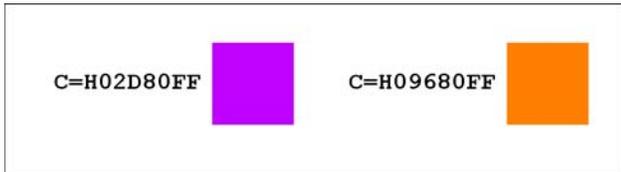
“turn the dial” between the successive relevant pairs of *hhh* values listed above. Below is some code to convert color wheel degrees into their hexadecimal codes for HLS hues, and to create color samples:

```

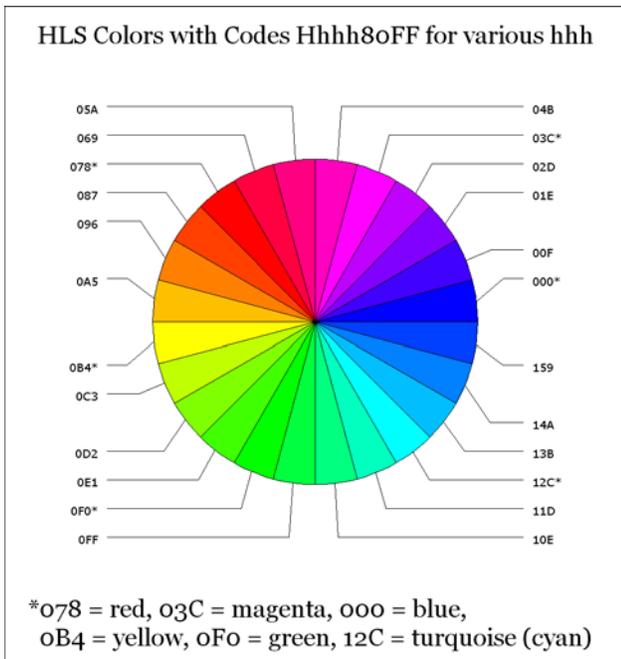
data _null_;
length HLScode $8.;
degrees = 45;
HLScode = 'H' || put(degrees,hex3.) || '80FF';
call symput('MyViolet',HLScode);
degrees = 150;
HLScode = 'H' || put(degrees,hex3.) || '80FF';
call symput('MyOrange',HLScode);
run;
goptions reset=all;
goptions device=PNG gsfname=anyname border;
goptions vpos=06 vsize=0.90 IN ymax=0.90 IN ypixels=270;
goptions hpos=34 hsize=3.25 IN xmax=3.25 IN xpixels=975;
filename anyname "YourDrive:\YourFile.png";
proc gslide;
title; footnote; note h=1 ' ';
note j=C          f='Courier New/Bold' h=1 c=H0000000 "C=&MyViolet"
      move=(+0.5,-1.25) f='Monotype Sorts' h=4 c=&MyViolet '6E'X
      move=(+2,+1.25) f='Courier New/Bold' h=1 c=H0000000 "C=&MyOrange"
      move=(+0.5,-1.25) f='Monotype Sorts' h=4 c=&MyOrange '6E'X;
run; quit;
filename anyname clear;

```

Here are the color samples for “My Violet” and “My Orange”:

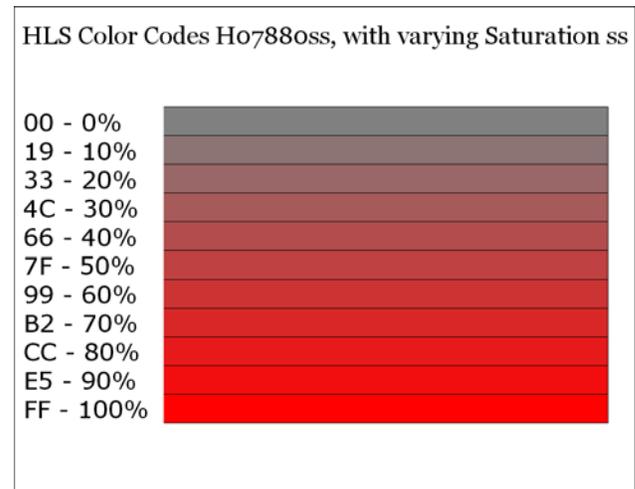
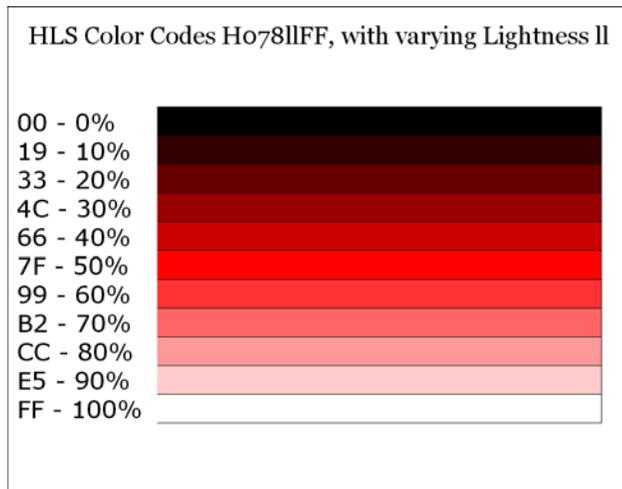


Here is what an HLS Color Wheel and a much denser, linearized version of it look like:



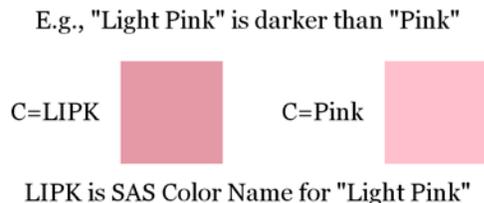
The narrow sharp “signals” at magenta, yellow, and turquoise also occur when the color wheel is created with minimal width slices. They are more pronounced when viewed on a monitor rather than printed.

Varying the lightness or the saturation, while holding other parameters constant, is demonstrated below.



### Verify What You Will Get from the Predefined Color Names and HTML Color Names

There are 292 “SAS Predefined Color Names”, listed in Table 7.2 in the Version 6 and Version 8 SAS/GRAPH documentation. They have names such as “PINK”, or “LIPK” for “Light Pink”. However, many colors differ from what you would expect based on their name. If you display or print PINK and LIPK, you will find that SAS Light Pink is darker than SAS Pink. See the illustration below. There are other situations like this. Also, many of the colors are too dark to be useful. Always make color samples.



Omitting GOPTIONS, here is the code for the above color sample:

```
proc gslide;
note j=C          f='Georgia'          h=1 c=CX000000 ' C=LIPK '
  move=(+1,-1.5) f='Monotype Sorts' h=5 c=LIPK      '6E'X
  move=(+3,+1.5) f='Georgia'          h=1 c=CX000000 'C=Pink'
  move=(+1,-1.5) f='Monotype Sorts' h=5 c=PINK      '6E'X;
run; quit;
```

If you cannot use Windows TrueType fonts, use f=CENTX, or some other SAS software font, and replace '6E'X with '03'X.

Also, there is another list of 144 SAS color names, in the SAS Color Registry. You can find them, and their RGB codes, using this click sequence in your Windows SAS session:

Solutions > Accessories > Registry Editor > Colornames > HTML

Of these color names, only 140 are HTML color names. Those 140 colors were originally developed for the Unix X Window system, and were later adopted as HTML color names. (Four of the colors in the SAS registry are SAS abbreviations for HTML Brown, HTML Green, HTML Orange, and HTML Purple. The registry also includes the four corresponding unabbreviated HTML color names.)

As with the previously discussed SAS predefined color names, for HTML color names, too, assume nothing. Make yourself a sample chart. The HTML names are not necessarily reasonable descriptions of the colors. E.g., NavajoWhite is not at all close to White. Though you could describe it as light orange, it is not a faintly orange-tinted white.

Only seven of the HTML color names intended for web use are browser-safe. Actually there are ten browser-safe HTML color names, but they include three pairs of colors that are two different names for the same RGB code.

**Ironically, the HTML color name standard palette is not web-safe.**

Also, there are three anomalies in the set of SAS HTML color names. PowderBlue, Turquoise, and PaleTurquoise are in the standard list of HTML colors, but are spelled differently in the SAS Color Registry. If you use the standard HTML name in an ODS program, you will get a message like the following in the SAS log:

```
WARNING: Possible unknown color: PowderBlue. Color will be passed
directly to output destination(s).
```

Fortunately, the resulting output does show the expected color—HTML recognizes it, but not SAS.

### Working with the SAS Color Registry

To get a printable listing in the SAS log, use this:

```
proc registry
startat='HKEY_SYSTEM_ROOT\COLORNAMES'
list;
run; quit;
```

To export the listing to a .txt file, use this:

```
proc registry
startat='HKEY_SYSTEM_ROOT\COLORNAMES'
export='C:\YourFolderName\YourFileName.txt';
run; quit;
```

Using the .txt file created with the code above as a template, you could create your own color list and import it back into the SAS Color Registry with a different color list name. That would enable you to use SAS software with your own custom palette, with RGB assignments that you like (e.g., browser-safe ones), and with your own names for them (presumably ones that you regard as reliably descriptive).

**NOTE: The long HTML color names can be used for ODS styles, or in the STYLE parameters with PROC PRINT, PROC REPORT, and PROC TABULATE. However, they cannot be assigned with C= in SAS/GRAPH, nor in any other ways that colors are specified in SAS/GRAPH.**

## Tools for Generating Large Numbers of Sample Color Combinations or Sample Colors

In Appendix B is laborsaving code that can be used to:

- (a) evaluate a large number of text (foreground) / background color combinations for readability; or
- (b) create a large number of color samples.

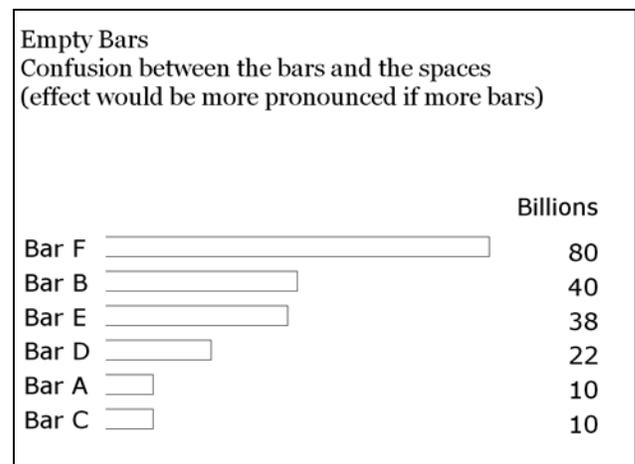
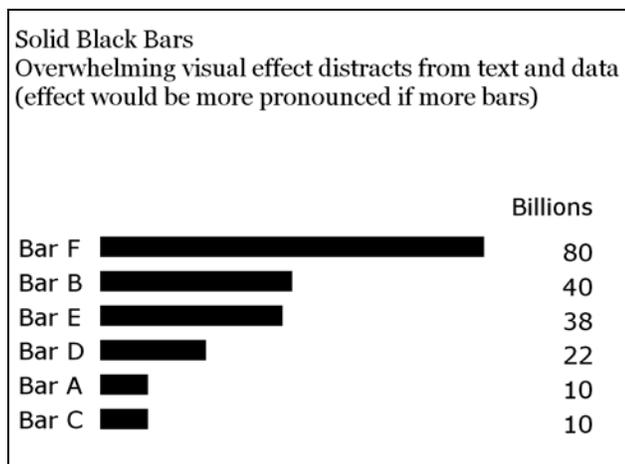
See Figures 6-9 for examples generated with these tools.

Except for the macro that is used to generate all 216 browser-safe colors, these tools (despite the fact that the macro parameters used to specify colors have the suffix “RGBcolor”) can actually be used with any other color that the SAS System recognizes: HLS colors, the SAS Predefined Color Names (such as TAN, CREAM, etc.), and any of the long HTML color names supported by SAS ODS.

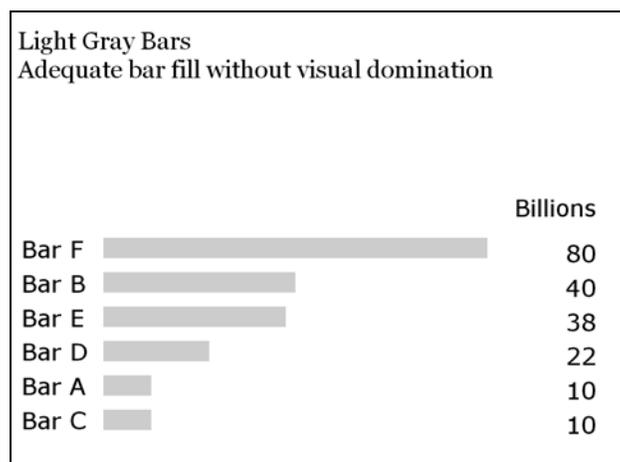
The examples do use a custom ODS style preferred by the author. You could substitute any ODS style that you prefer, but your web page background should be white, so as to not affect the visual perception of the colors being evaluated/sampled.

### A Light Color May Be the Right Color

With many more bars in the example below, the effects of too much or too little color would be worse than what you can see below.



Below, a light color creates solid area fill without dominating the visual and information messages.



## Color Excess: All of the Places To Apply ODS STYLE Parameters in PROC PRINT

The image shows two browser windows displaying PROC PRINT output tables. The left window shows a table with columns 'Age', 'Name', and 'Height'. The right window shows a table with columns 'Obs', 'Age', 'Name', and 'Height'. Both tables use various colors and fonts for headers, data rows, and totals.

Age	Name	Height
13	Alice	56.5
	Barbara	65.3
13		121.8
N = 2		
14	Carol	62.8
	Judy	64.3
14		127.1
		248.9
N = 2 Total N = 4		

Obs	Age	Name	Height
1	13	Alice	56.5
2	13	Barbara	65.3
3	14	Carol	62.8
4	14	Judy	64.3
			248.9
N = 4			

One can control the color and other display/format aspects of a table by creating a custom ODS style, but for maximum flexibility one can apply the controls directly inside the reporting procedure code. Listed below is the code used to create the above demonstration tables with PROC PRINT. Similar controls are available for PROC TABULATE and PROC REPORT.

```
proc sort data=sashelp.class out=ToPrint;
where name in ('Alice' 'Barbara' 'Carol' 'Judy');
by Age Name;
run;

%let FontFormatting = %str(font_weight=Bold font_size=6);

ods noresults;
ods listing close;
ods html path='C:\' (url=none)
body='STYLEinPROCPRINToutput_WithIDvar.html'
(title='All The Places To Apply ODS STYLE= Except OBS and OBS Header')
style=styles.Minimal;
title;
proc print data=ToPrint label N
style(header) = [&FontFormatting]
style(data) = [&FontFormatting]
style(total) = [&FontFormatting background=magenta foreground=cyan]
style(grandtotal) = [&FontFormatting background=white foreground=black]
style(N) = [&FontFormatting background=cyan foreground=magenta just=left];
by age;
id age /
```

```

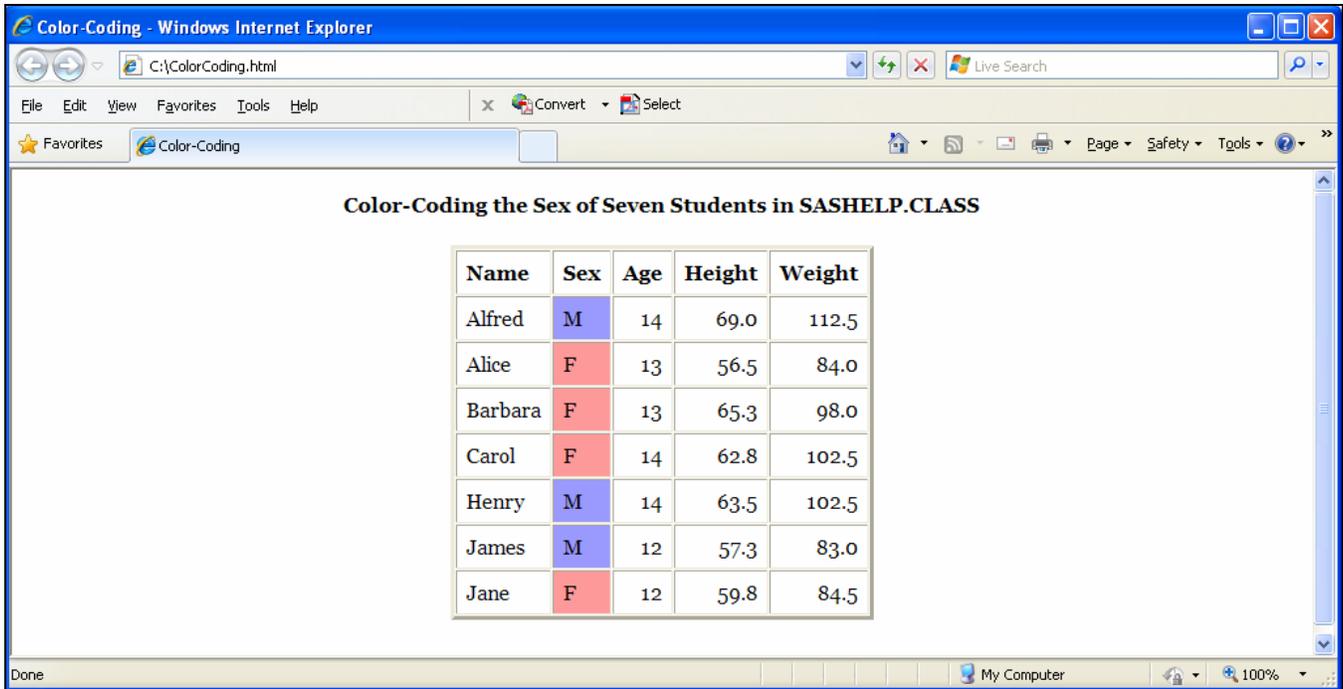
style(header) = [&FontFormatting background=blue foreground=red]
style(data)   = [&FontFormatting background=red foreground=blue];
var Name /
style(header) = [background=black foreground=CX999999]
style(data)   = [background=CX999999 foreground=black];
var Height;
sum Height /
style(header) = [background=CX009900 foreground=yellow]
style(data)   = [background=yellow foreground=CX009900]
style(total)  = [&FontFormatting background=black foreground=white];
sumby age;
run;
ods html close;

ods html path='C:\' (url=none)
body='STYLEinPROCPRINToutput_WithOBSvar.html'
(title='All The Places To Apply ODS STYLE= Except ID and ID Header')
style=styles.Minimal;
title;
proc print data=ToPrint label N
style(header) = [&FontFormatting]
style(data)   = [&FontFormatting]
style(obsheader) = [&FontFormatting background=brown foreground=orange]
style(obs)     = [&FontFormatting background=orange foreground=brown]
style(total)  = [&FontFormatting background=magenta foreground=cyan]
style(grandtotal) = [&FontFormatting background=white foreground=black]
style(N)      = [&FontFormatting background=cyan foreground=magenta just=left];
var age /
style(header) = [&FontFormatting background=blue foreground=red]
style(data)   = [&FontFormatting background=red foreground=blue];
var Name /
style(header) = [background=black foreground=CX999999]
style(data)   = [background=CX999999 foreground=black];
var Height;
sum Height /
style(header) = [background=CX009900 foreground=yellow]
style(data)   = [background=yellow foreground=CX009900]
style(total)  = [&FontFormatting background=black foreground=white];
run;
ods html close;
ods listing;

```

## Color-Coding Your Data (NOT “Traffic-Lighting”)

As previously mentioned the regrettably popular fascination with traffic-lighting of data is non-communicative for color-blind viewers. Here the colors used for coding are light red and light blue for female and male, respectively.



The screenshot shows a web browser window with the following content:

**Color-Coding the Sex of Seven Students in SASHELP.CLASS**

Name	Sex	Age	Height	Weight
Alfred	M	14	69.0	112.5
Alice	F	13	56.5	84.0
Barbara	F	13	65.3	98.0
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83.0
Jane	F	12	59.8	84.5

Here is the code used:

```
proc format;
value $SexCol
    'F'='CXFF9999'
    'M'='CX9999FF';
run;

ods noresults;
ods listing close;
ods html path='C:\' (url=none)
    body='ColorCoding.html'
    (title='Color-Coding')
style=styles.Minimal;
title font='Georgia/Bold'
    'Color-Coding the Sex of Seven Students in SASHELP.CLASS';
options obs=7;
proc print data=sashelp.class label noobs
    style(header) = [font_face='Georgia']
    style(data) = [font_face='Georgia'];
var Name;
var Sex / style(data) = [background=$SexCol.];
var Age Height Weight;
run;
options obs=max;
ods html close;
ods listing;
```

## Color Differs on Different Media. Do You See What I See?

My wife and I disagree on whether certain colors are green (what I see) or brown (what she sees). But there are more than mere differences in human visual perception. In addition to what can happen when using browser-unsafe colors on the web, and what are called “gamma differences” between PC, Mac, and Unix, there are other technology-related sources of variation. Here are some of them:

- CRT monitor color and LED flat panel color differ.
- On an LED panel, very light colors wash out to near-White.
- LED projector color differs from color on the presenter’s PC or laptop that feeds the projector.
- CRT or LED color differs from printer color.
- Hardcopy color varies from printer to printer.

Among my experiences in color communication was to see an LED projector convert blue and red text on my PowerPoint slides into violet and orange. This is more than the color difference phenomena mentioned above. The LED projector is probably the riskiest color communication tool. Tuning a shared projector to suit one’s own laptop is time-consuming, and may impair the usefulness of the projector for some other presenter.

### Conclusion

Color is something that we take for granted. However, without getting into details about the physiology of color perception, optical illusions due to color perception, and other arcane subtleties, this paper has shown that color selection requires care if we want to get beyond mere decoration and into effective communication, and has provided guidelines as well as color evaluation tools.

### Commented List of References

1. Bessler, LeRoy (1995), “Communicate Effectively in Color with SAS/GRAPH Software”, *Proceedings of the Twentieth Annual SAS Users Group Conference*, Cary, NC, USA: SAS Institute Inc., 1995. This was the first SAS users group conference paper on communication-effective use of color.
2. Bessler, LeRoy (2004), “Communication-Effective Use of Color for Web Pages, Graphs, Tables, Maps, Text, and Print”, *Proceedings of the Twenty-Ninth Annual SAS Users Group Conference*, Cary, NC, USA: SAS Institute Inc., 2004. Find it on the web at <http://www2.sas.com/proceedings/sugi29/176-29.pdf>. It is the source of this updated and enhanced MWSUG 2010 paper that you are reading.

### Author Information

Your questions, comments, and ideas about communicating with color are always welcome.

LeRoy Bessler PhD

Bessler Consulting and Research, Fox Point, Milwaukee, Wisconsin, USA

[Le\\_Roy\\_Bessler@wi.rr.com](mailto:Le_Roy_Bessler@wi.rr.com)

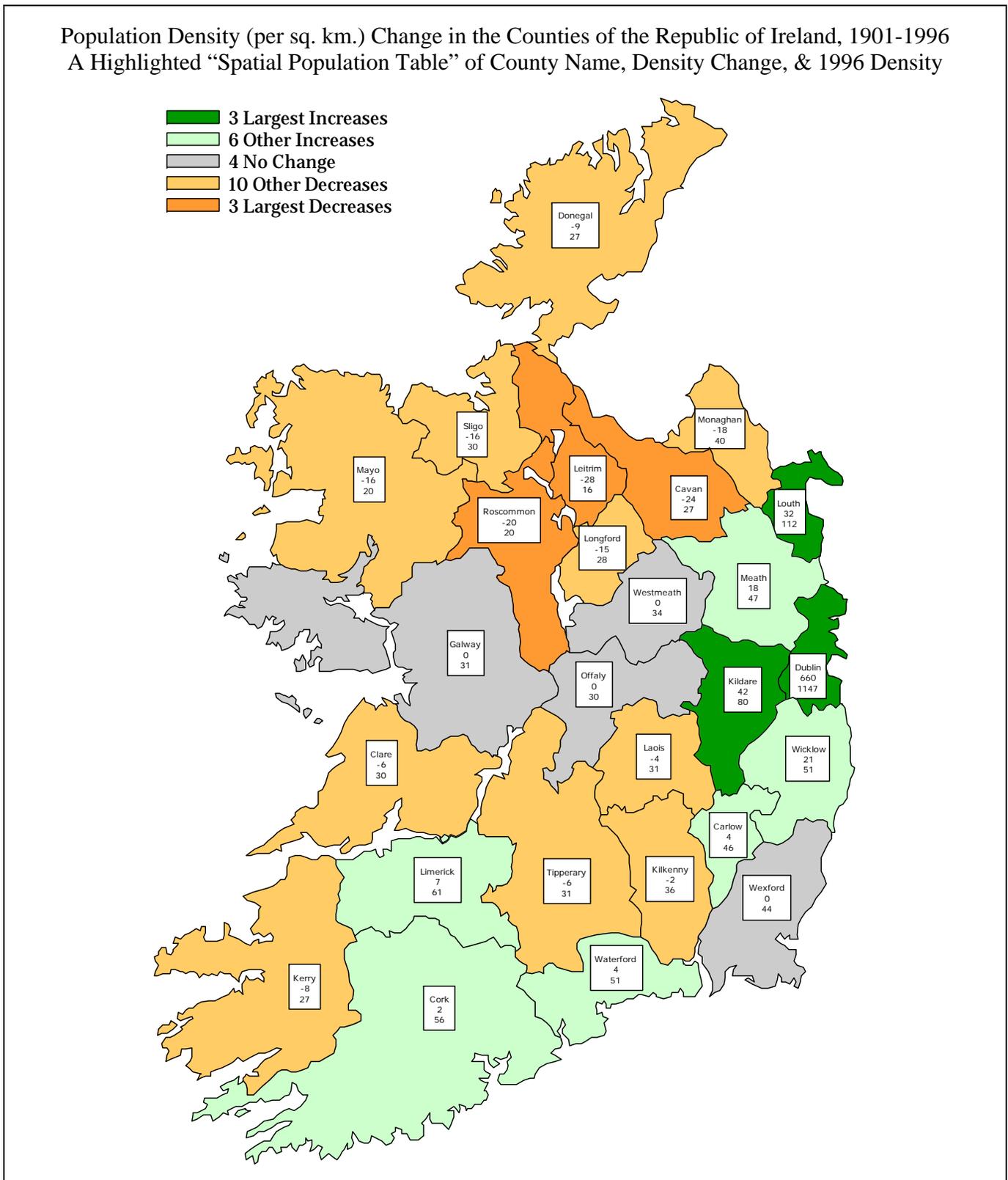
A SAS user since 1978, Dr. LeRoy Bessler has shared his knowledge and experience with other users at conferences throughout the USA and in Montreal, London, Heidelberg, and Dublin. Though a SAS generalist with long experience in Base SAS, SAS macro language, and SAS tools for access to non-SAS data, his special interests include communication-effective visual communication and reporting, web information delivery, highly formatted Excel reporting, SAS/GRAPH®, ODS, creation of unique tools to support the SAS BI server and its users, and Software-Intelligent Application Development for Reliability, Reusability, Extendibility, and Maintainability. He is a regular contributor to *VIEWS News*, the web newsletter of the VIEWS International SAS Programmer Community.

SAS is a registered trademark or trademark of SAS vendor Inc. in the USA and other countries.

® indicates USA registration. Other product and brand names are trademarks or registered trademarks of their respective owners.

# Appendix A.

Figure 1. An Alternative to Traffic Lighting: augmented with light shades of the signal colors.



Shown at the Eighteenth Annual SAS European Users Group International Conference, Dublin, 2000

Figure 2. Samples of Browser-Safe SAS/GRAPH Colors, with Their RGB Codes



Figure 3. Adding Gray to Browser-Safe Primaries and Secondaries



Figure 4. Some Other Sets of “Related” Browser-Safe Colors

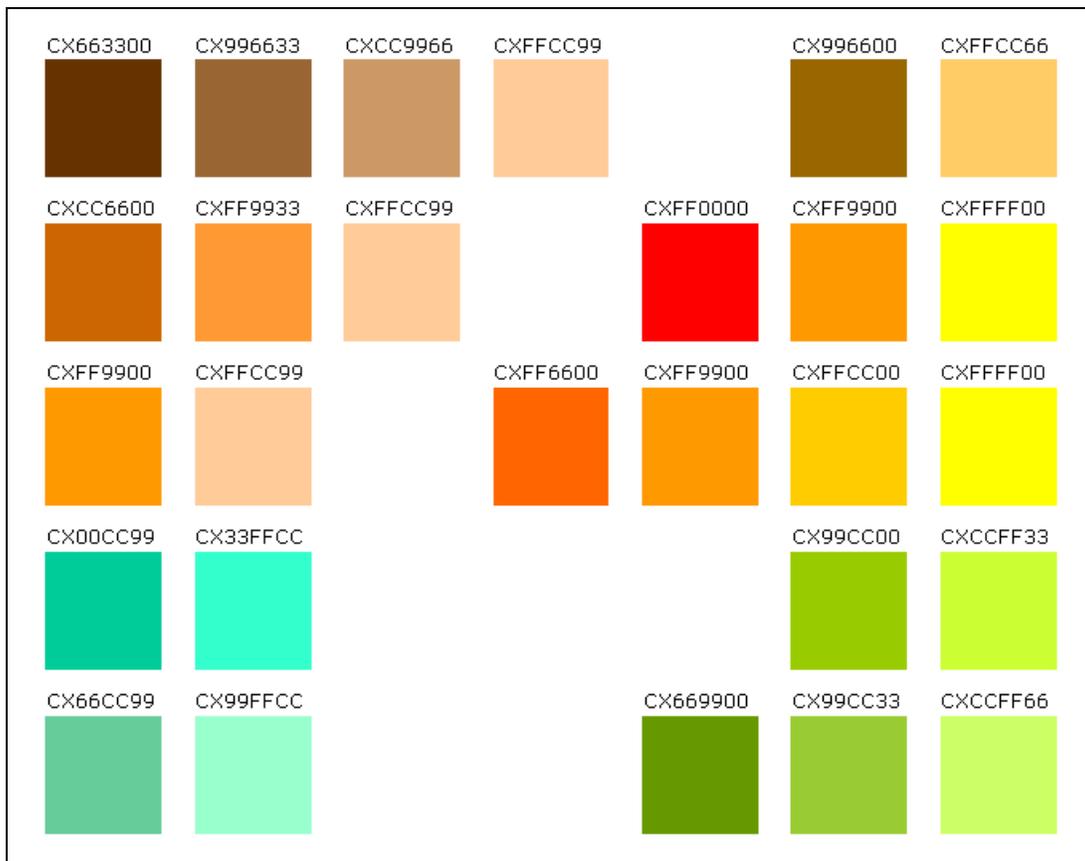


Figure 5. The 216 Browser-Safe Colors with Their RGB Codes

000000	000033	000066	000099	0000CC	0000FF	003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF	009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF	00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF	333300	333333	333366	333399	3333CC	3333FF
336600	336633	336666	336699	3366CC	3366FF	339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF	33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF
660000	660033	660066	660099	6600CC	6600FF	663300	663333	663366	663399	6633CC	6633FF
666600	666633	666666	666699	6666CC	6666FF	669900	669933	669966	669999	6699CC	6699FF
66CC00	66CC33	66CC66	66CC99	66CCCC	66CCFF	66FF00	66FF33	66FF66	66FF99	66FFCC	66FFFF
990000	990033	990066	990099	9900CC	9900FF	993300	993333	993366	993399	9933CC	9933FF
996600	996633	996666	996699	9966CC	9966FF	999900	999933	999966	999999	9999CC	9999FF
99CC00	99CC33	99CC66	99CC99	99CCCC	99CCFF	99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF	CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF	CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF	CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF	FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF
FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF	FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF
FFCC00	FFCC33	FFCC66	FFCC99	FFCCCC	FFCCFF	FFFF00	FFFF33	FFFF66	FFFF99	FFFFCC	FFFFFF

Figure 6. Bad Examples of Text-Background Color Combinations

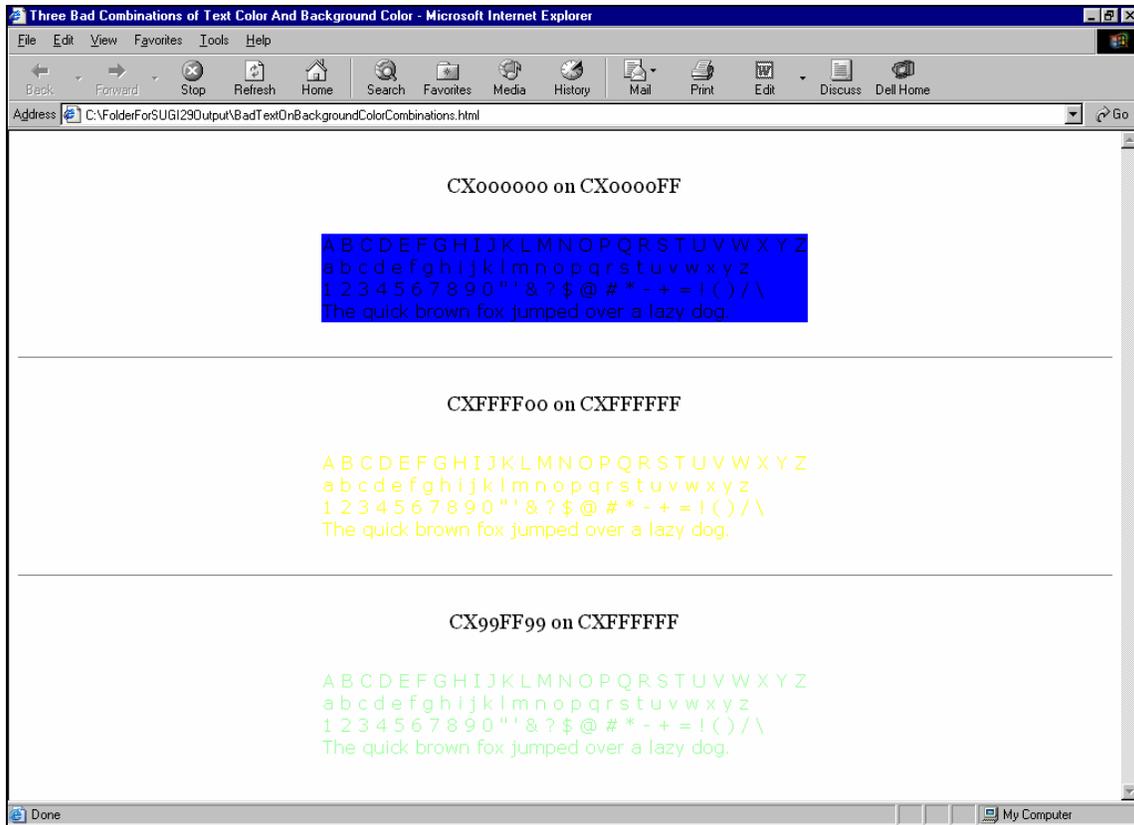


Figure 7. Good Examples of Text-Background Color Combinations

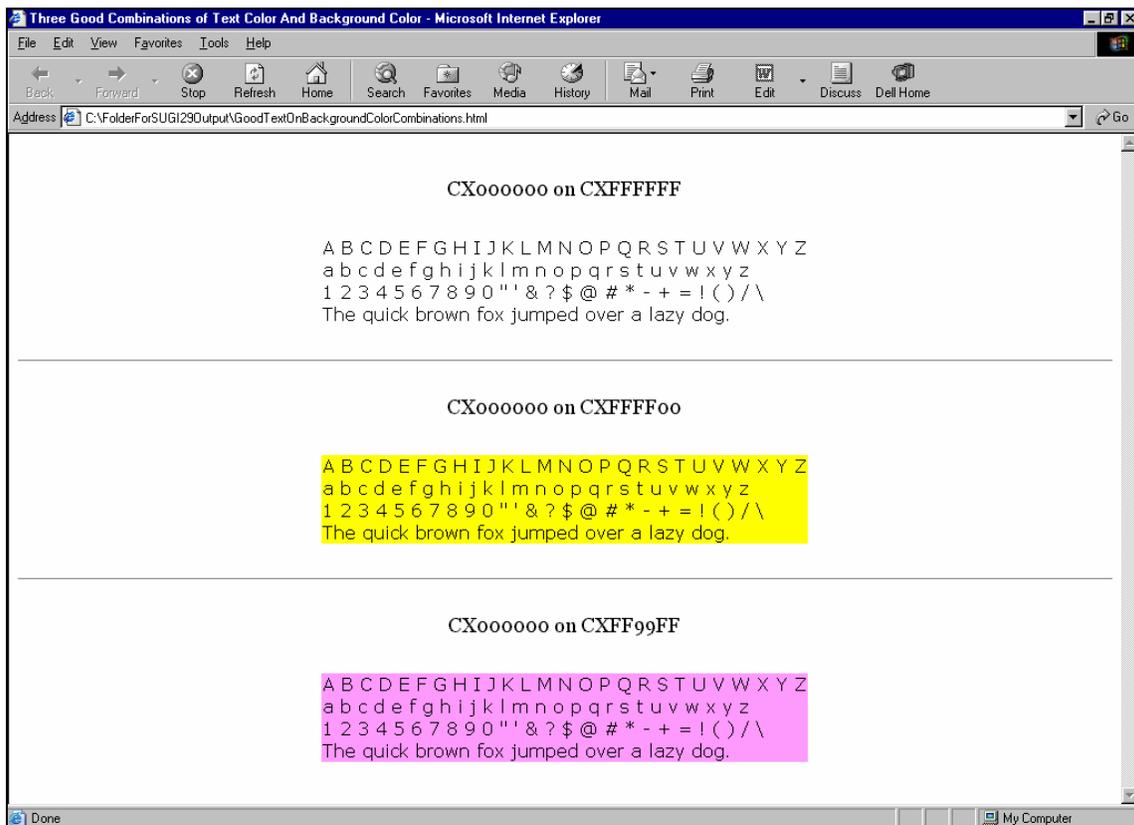


Figure 8. SAS Colors Blue, Tan, & Cream, with My Monitor Set to 32-bit Color Mode

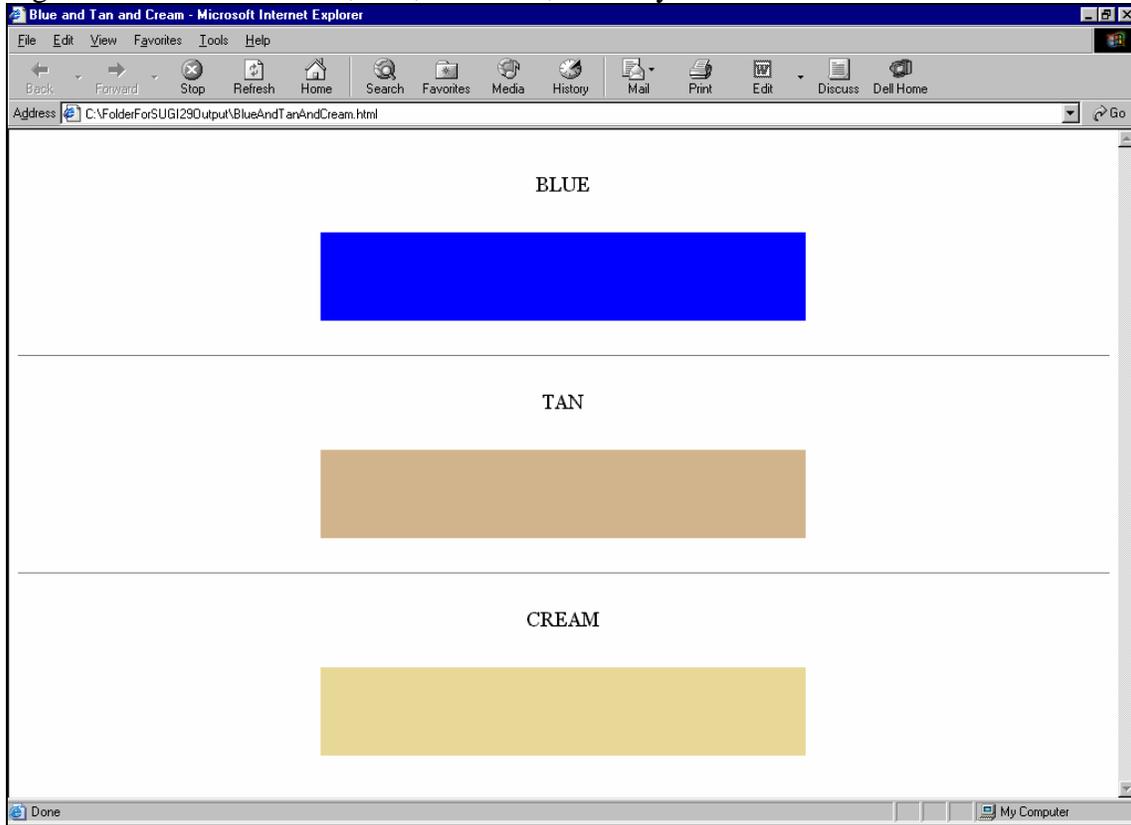
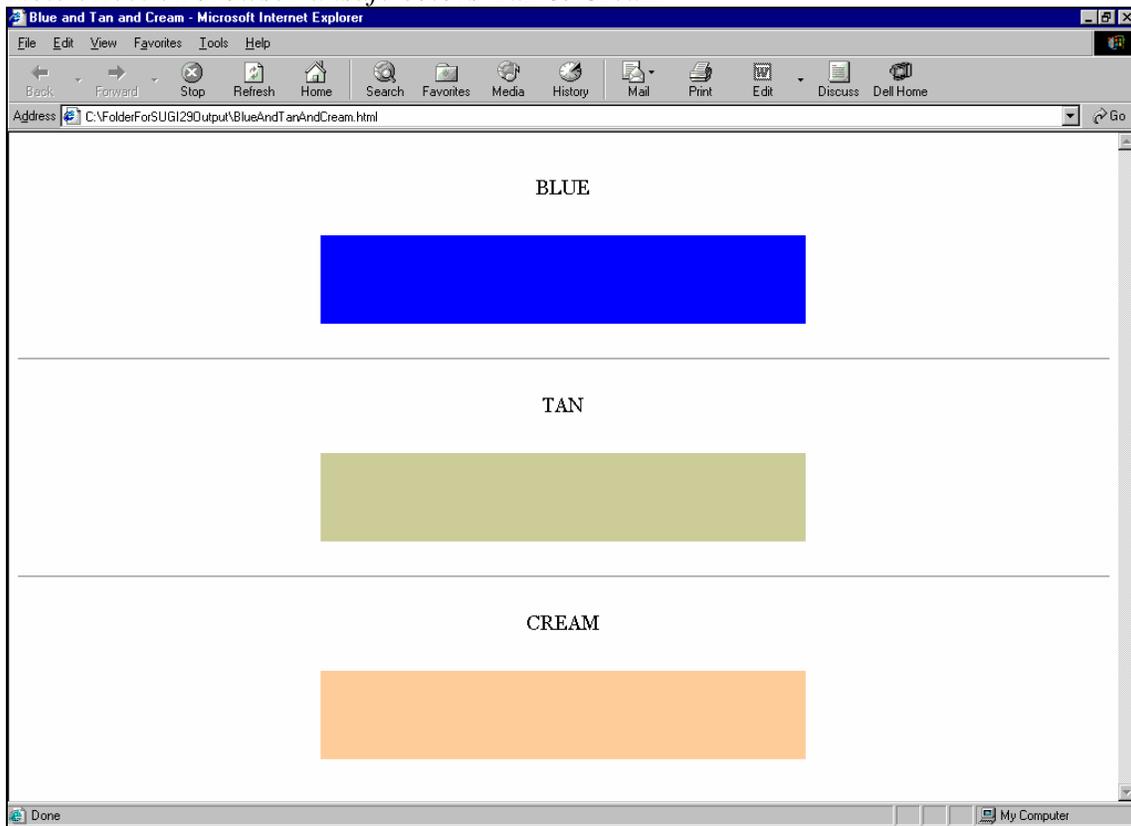


Figure 9. SAS Colors Blue, Tan, & Cream, with My Monitor Set to 256-Color Mode  
Note effect on *browser-unsafe colors* Tan & Cream



# Appendix B. Macro Tools and Sample Programs for Evaluating Colors and Combinations

## Macro To Create a Custom ODS Style

NOTE: The macro parameters used to specify colors, despite the fact that they have the suffix "RGBcolor", can actually be assigned any color that the SAS System recognizes.

```
%macro CustomBaseStyleBuild(
StyleName=LeRBSugi29CustomStyle,
PROCOutputSeparators=NO, /* YES to put horizontal rule
between successive PROC outputs in the same web page,
but would have NO EFFECT if using NEWFILE = PROC. */
PROCOutputSepLineThickness=1, /* 2 is thicker,
3 is Default */
WebPageBackgroundRGBcolor=CXFFFF99,
/* CXFFFF99 is light Web-Safe yellow */
TitleFootnoteBackgroundRGBcolor=CXFFFF99,
TitleFootnoteBackgrdTransparency=NO, /* YES to let
web page background show through */
TitleFootnoteRGBcolor=CX000000, /* Web-Safe black */
TitleFootnoteFont=Georgia,
TitleFootnoteSize=4,
TableBackgroundRGBcolor=CXFFFF99,
TableContentRGBcolor=CX000000, /* data and headings */
TableHeadingFont=Verdana,
TableHeadingSize=3,
TableDataFont=Verdana,
TableDataSize=3,
TableFrame=box, /* TableFrame=void to remove frame */
TableFrameRulesGridRGBcolor=CX9999FF,
/* light Web-Safe blue */
TableGrid=NO, /* YES for a grid between data cells */
TableSpacing=5); /* the SAS-shipped default is 7.
This is space between cell data & cell boundaries. */

proc template;

edit styles.Default as styles.&StyleName;

/* Create a modified style based
on the ODS STYLES.DEFAULT.
Anything not referenced or overridden here
will be controlled by the ODS Default Style. */

style fonts /

'TitleFont' =
("&TitleFootnoteFont, Times New Roman, Times",
&TitleFootnoteSize)
/* "system" titles & footnotes */

'HeadingFont' =
("&TableHeadingFont, Times New Roman, Times",
&TableHeadingSize)
/* column & row headings, and OBS, ID, & SUM values */

'DataFont' =
("&TableDataFont, Arial, Helvetica",
&TableDataSize)
/* table data. DataFont being added by LeRB.
Not in ODS Default style. */

'DocFont' = ("Comic Sans MS, Courier",4);
/* My default for unassigned fonts.
Conspicuous font chosen to be obvious if used
by ODS, so that a way can be found to assign a
preferred font, instead of "my default". */

style color_list /

'SafeRed' = CXFF0000 /* Browser-Safe red */
'SafeBlue' = CX0000FF /* Browser-Safe blue */
'SafeMagenta' = CXFF00FF /* Browser-Safe magenta */

'WebPageBackgroundRGBcolor' =
&WebPageBackgroundRGBcolor

'TitleFootnoteBackgroundRGBcolor' =
&TitleFootnoteBackgroundRGBcolor

'TitleFootnoteColor' =
&TitleFootnoteRGBcolor
```

```
'TableBackgroundRGBcolor' =
&TableBackgroundRGBcolor

'TableContentColor' =
&TableContentRGBcolor

'TableBoundariesColor' =
&TableFrameRulesGridRGBcolor;

style colors /
'systitlefg' =
color_list('TitleFootnoteColor')
/* "system" titles & footnotes */
'systitlebg' =
color_list('TitleFootnoteBackgroundRGBcolor')
/* background for "system" title/footnote areas
However, if transparency is enabled,
then this color is ignored. */
'headerfg' =
color_list('TableContentColor')
/* override fgA2, which is the
ODS default for table row & column labels */
'headerbg' =
color_list('TableBackgroundRGBcolor')
/* background for table row & column labels */
'datafg' =
color_list('TableContentColor')
/* table cell data */
'databg' =
color_list('TableBackgroundRGBcolor')
/* background for table cell data */
'docfg' =
color_list('SafeMagenta')
/* My default for unassigned foreground colors.
Conspicuous color chosen to be obvious if used
by ODS, so that a way can be found to assign a
preferred color, instead of "my default". */
'docbg' =
color_list('WebPageBackgroundRGBcolor')
/* background for web page and ??? */
'tableborder' =
color_list('TableBoundariesColor')
/* actually, for table frame AND table rules */
'TableGrid' =
color_list('TableBoundariesColor')
/* (TableGrid is an LeRB replacement for where
tablebg is used by ODS default style)
Color of table grid when cellspacing > 0 AND
"style Table from Output"
does not assign background. */
'link2' =
color_list('SafeBlue') /* std for unvisited links */
'link1' =
color_list('SafeRed'); /* std for visited links */

style SysTitleAndFooterContainer from Container /
cellpadding = 0 /* compact the title/footnote area */
cellspacing = 0 /* no grid for title/footnote area */
%if %upcase(&TitleFootnoteBackgrdTransparency) = YES
%then %do;
background = _undef_;
style systitle / background = _undef_;
style systemfooter / background = _undef_;
/* Three instances of background = _undef_ above
make the title and footnote areas transparent.
They let the web page background show through.
When this option is selected,
the systitlebg color is actually ignored. */
%end;
%else %do;
; /* needed to end this STYLE statement */
%end;

style Output from Container /
/* these statements control
table grid and table border */
rules = NONE /* NONE to override rules=GROUPS,
preventing double line between table labels & data.
ALL would create fixed-width thin line
around all interior cells and
at inner edges of all perimeter cells */
%if %upcase(&TableGrid) eq NO
```

```

%then %do;
  frame = &TableFrame /* BOX for on, VOID for off */
  cellspacing = 0 /* override 1. Space between cells.
  Also, space between outer cells and any frame. */
%end;
%else %do;
  frame = VOID /* but keeping default cellspacing=1 */
%end;
cellpadding = &TableSpacing /* override default 7 */
background = colors('TableGrid')
  /* color of grid (LeRB replacement for tablebg),
  if cellspacing > 0 AND
  "style Table from Output" does not override it.
  This is NOT the table background on the web page. */
bordercolor = colors('tableborder')
  /* color of table frame and table rules */
borderwidth = 1;
  /* thickness of table frame, same as default */
/* NOTE: bordercolor affects more table parts
than does borderwidth. There is no apparent
way to thicken the rules, when present. */

style Data from Cell / font = fonts('DataFont');
  /* Added to override default use of DocFont */

%if %upcase(&PROCoutputSeparators) = NO %then %do;
style Body / pagebreakhtml = _undef_;
  /* suppress rule between successive PROC outputs */
%end;
%else %do;
style html / 'ThinLineAfterSpace' =
  "&#160;<hr size=&PROCoutputSepLineThickness>";
style Body / pagebreakhtml =
  html('ThinLineAfterSpace');
  /* one space and a thin line
  between successive PROC outputs */
%end;

end; run; quit;

%mend CustomBaseStyleBuild;

```

## Creating the Custom ODS Style

```

%CustomBaseStyleBuild(
  StyleName=LeRBSugi29ColorDemo,
  TableHeadingSize=1,
  TableFrame=void,
  TableSpacing=1,
  WebPageBackgroundRGBcolor=CXFFFFFFF,
  /* Web-Safe white */
  TableBackgroundRGBcolor=CXFFFFFFF,
  TitleFootnoteBackgrdTransparency=YES,
  PROCoutputSeparators=YES)

run;

```

## Data for Color Text Displays

```

data FontCharacters; label FontCharacters='00'X;
infile cards; input @1 FontCharacters $51.;
cards;
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
1 2 3 4 5 6 7 8 9 0 " ' & ? $ @ # * - + = ! ( ) / \
The quick brown fox jumped over a lazy dog.
;
run;

```

## Macro to Display Text on a Background or to Create a Solid Color Sample

```

%macro ColorTable(TextColor=, BackgroundColor=);
title1 "&TextColor"
%if %upcase(&BackgroundColor) ne %upcase(&TextColor)
%then %do;
  " on &BackgroundColor"
%end;

proc print data=FontCharacters noobs label
  style(data)=[foreground=&TextColor
              background=&BackgroundColor];
run;
%mend ColorTable;

```

## Creating Text-Background Color Combinations and Solid Color Samples (Figures 6, 7, 8, & 9)

```

ods listing close; ods noresults; goptions reset=all;
ods html
  path = "C:\FolderForSUGI29Output" (url=None)
  body = "BadTextOnBackgroundColorCombinations.html"
  (title="A Few Bad Combinations of Text Color And
  Background Color")
  style = Styles.LeRBSugi29ColorDemo
  newfile = NONE; * one continuous scrollable
  web page body file *;
%ColorTable(TextColor =CX000000,
  BackgroundColor=CX0000FF)
run;
%ColorTable(TextColor =CXFFFF00,
  BackgroundColor=CXFFFFFFF)
run;
%ColorTable(TextColor =CX99FF99,
  BackgroundColor=CXFFFFFFF)
run;
ods html
  path = "C:\FolderForSUGI29Output" (url=None)
  body = "GoodTextOnBackgroundColorCombinations.html"
  (title="A Few Good Combinations of Text Color And
  Background Color")
  style = Styles.LeRBSugi29ColorDemo newfile = NONE;
%ColorTable(TextColor =CX000000,
  BackgroundColor=CXFFFFFFF)
run;
%ColorTable(TextColor =CX000000,
  BackgroundColor=CXFFFFFF00)
run;
%ColorTable(TextColor =CX000000,
  BackgroundColor=CXFF99FF)
run;
ods html
  path = "C:\FolderForSUGI29Output" (url=None)
  body = "BlueAndTanAndCream.html"
  (title="Blue and Tan and Cream")
  style = Styles.LeRBSugi29ColorDemo newfile = NONE;
%ColorTable(TextColor =BLUE,
  BackgroundColor=BLUE)
run;
%ColorTable(TextColor =TAN,
  BackgroundColor=TAN)
run;
%ColorTable(TextColor =CREAM,
  BackgroundColor=CREAM)
run;
ods html close; ods listing;

%macro AllBrowserSafeColors;
%do RRdecimal = 0 %to 255 %by 51;
  %do GGdecimal = 0 %to 255 %by 51;
    %do BBdecimal = 0 %to 255 %by 51;
data _null_;
DecimalCode = &RRdecimal;
call symput('RR',put(DecimalCode,hex2.));
DecimalCode = &GGdecimal;
call symput('GG',put(DecimalCode,hex2.));
DecimalCode = &BBdecimal;
call symput('BB',put(DecimalCode,hex2.));
run;
%ColorTable(TextColor =CX&RR&GG&BB,
  BackgroundColor=CX&RR&GG&BB)
run;
%end;
%end;
%end;
%macro AllBrowserSafeColors;
ods listing close; ods noresults; goptions reset=all;
ods html
  path = "C:\FolderForSUGI29Output" (url=None)
  body =
  "AllOfTheBrowserSafeColors.html" (title=" . . . ")
  style = Styles.LeRBSugi29ColorDemo newfile = NONE;
%AllBrowserSafeColors
run;
ods html close; ods listing;

```