ABSTRACT
The purpose of this paper is to demonstrate how PROC FORMAT can be utilized to produce a standard coding mechanism when analyses based upon timeframe windows are required. This technique incorporates a series of two complimentary formats (i.e., one containing a sequence of value ranges and one containing the corresponding labels for each sequence of value ranges) which are used to create macro variables for use in data creation and building tables and listings. Using this technique is one way to ensure that standard timeframes required by your project are maintained.

The benefits of such a mechanism are:

1. It establishes the building blocks, namely macro fields, for generating dynamic and consistent code within each program based upon expected values of the data
2. It allows for porting consistent code across programs
3. It eliminates the need to update programs if timeframe ranges change or additional timeframes are required

INTRODUCTION
Many studies collect data in a longitudinal fashion. Many times this makes it necessary to group events according to when they occur and to view them in relation to other events occurring at relatively the same or different times. These instances require defining Analysis Windows (timeframes) and providing labels for them. The consistent nature and use of this timeframe / label combination for the derivation and reporting of most if not all analysis fields (e.g., lab / other test results, clinical outcomes, compliance, Adverse Events (AEs), etc.) lends itself to a standard coding mechanism for establishing timeframe parameters and their corresponding labels.

In addition to improved consistency, greater efficiency is also achieved. When changes to specifications occur that include a request for additional or modified timeframes, it is beneficial to have a mechanism in place which minimizes, if not eliminates, the need to modify existing programs.

This paper will demonstrate a technique, which uses the information contained in SAS formats, to establish such a mechanism, and centralize the definition of timeframes and their corresponding labels.

NOTE: The approach demonstrated in this paper was developed using SAS® Version 9.2, but is applicable to all versions of SAS® currently in use.

ANALYSIS WINDOWS
In this example we will use Analysis Windows in which days is the unit of measurement. Such an approach is typical for an adverse or study event report which group events according to when they were observed. Our reporting will be based upon ranges of days, with the Day of Treatment (drug environment) or Day of Procedure (device environment) being reported as a single day range (usually DAY 0 or DAY 1 – depending upon your standard convention). I will use DAY 0 in the context of this paper.

The value of DAY is based upon the number of days prior to or after the Date of Treatment or Procedure in which a milestone occurs. For example:

<table>
<thead>
<tr>
<th>MILESTONE</th>
<th>DATE</th>
<th>DAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treatment</td>
<td>12OCT2006</td>
<td>0</td>
</tr>
<tr>
<td>Pre-Treatment Blood Labs</td>
<td>10OCT2006</td>
<td>-2</td>
</tr>
<tr>
<td>Week 1 Post-Treatment Blood Labs</td>
<td>16OCT2006</td>
<td>4</td>
</tr>
<tr>
<td>Week 2 Post-Treatment Blood Labs</td>
<td>22OCT2006</td>
<td>10</td>
</tr>
<tr>
<td>Week 3 Post-Treatment Blood Labs</td>
<td>30OCT2006</td>
<td>18</td>
</tr>
</tbody>
</table>
Usually, the Analysis Windows are defined in advance, prior to conducting a clinical trial study, in the Study Protocol and the Statistical Analysis Plan. Given the advance nature of this information, you can establish formats ahead of time to address your Analysis Windows.

ANALYSIS WINDOW FORMATS
For the purpose of this paper we will use the following formats:

STUDYPD – provides the day range for each analysis window

STDYYPDF – provides the label for each Analysis Window (day range window)

```
proc format library=library;
value studypd
  -5 - -1 = 1
  0 = 2
  1 - 7 = 3
  8 - 16 = 4
  17 - 36 = 5;
quit;
```

```
proc format library=library;
value studypdf
  1 = 'Pre-Treatment'
  2 = 'Treatment'
  3 = '1 Week'
  4 = '2 Weeks'
  5 = '1 Month';
quit;
```

There are a couple of items to note here:

- The formats are complimentary in that there is a 1-to-1 correspondence between the values for each
- The range of day values in STUDYPD are continuous and contiguous – necessary for accounting for all of the days within each timeframe window and allowing for the windows to cover the full range of days necessary.
- The values on the right hand side of the ‘=’ sign for STUDYPD are numbered consecutively, starting with ‘1’.
- Once the code for each of these formats is submitted, the formats will be stored in a format library
- The code shown above stores the formats in a permanent format library (LIBRARY). This allows each format to be available for all programs accessed during a SAS Session or any program which references this library.

CHECKING YOUR FORMATS
There is an option in PROC FORMAT which allows you to review the formats you have created. The following code generates a listing that describes the values contained in the two formats just created (in the designated Format Library - Library). As you can see, the values reported correspond to the values specified in the statements above.

```
proc format library=library fmtlib;
select studypd studypdf;
run;
```
ACCESSING THE VALUES STORED IN FORMATS – ESTABLISHING A STANDARD CODING MECHANISM

SAS® has an option in PROC FORMAT that can be used to access the values contained within a format and output this information into a SAS Dataset.

The `CNTLOUT` option is used to generate a SAS Dataset based upon the values stored in a format residing in a SAS Format Library:

```sas
proc format library=library cntlout=filename;
  select formatname;
run;
```

The above code allows for access to the information stored in the format `formatname` and then stores this information in a SAS dataset `filename`. You can then use dataset `filename` to build an array of macro fields which can then be used within a program / macro to dynamically process data -- a standard coding mechanism.

In terms of the two formats we are using:

<table>
<thead>
<tr>
<th>Format: STUDYPD</th>
<th>FORMAT NAME: STUDYPD LENGTH: 1 NUMBER OF VALUES: 5</th>
<th>MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 1 FUZZ: STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>END</td>
<td>LABEL (VER. V7</td>
</tr>
<tr>
<td>-6</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>17</td>
<td>35</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>END</td>
<td>LABEL (VER. V7</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1 Pre-Treatment</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2 Treatment</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>3 1 Week</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>4 2 Weeks</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>5 1 Month</td>
</tr>
</tbody>
</table>
CONVERTING THIS INFORMATION INTO SAS MACRO FIELDS

The following SAS code can be used to read in the data from WORK.STUDYPD and WORK.STUDYPDF to create the macro fields which will serve as the building blocks for the standard coding mechanism we are trying to achieve.

**STEP#1:** Determine the number of Analysis Windows (macro field: NCOLPDS)

```sas
%global ncolpds;
%let ncolpds = 0;

data _null_;  
if(0)then set studypd point = _n_ nobs = count;  
   call symput('ncolpds',trim(left(count)));  
stop;  
run;
```

Notes:

- The resultant macro field NCOLPDS can be used to generate additional macro fields which store the ranges and labels for each Analysis Window.
- We are using the %GLOBAL2 command here and in the following steps to make these macro variables are available during the entire execution of the SAS session or job.

**STEP#2:** Build the ranges (beginning and ending days) for each Analysis Window (macro fields: BEG# and END#, respectively, where # is number 1 through the value stored in macro field: NCOLPDS)

```sas
data _null_;  
set studypd;
   %do col = 1 %to &ncolpds;
      %global beg&col end&col;
      if(label = &col)then do;
         call symput("beg&col",trim(left(start)));
         call symput("end&col",trim(left(end)));
      end;
   %end;
run;
```

### Table: STU&DYD

<table>
<thead>
<tr>
<th>START</th>
<th>END</th>
<th>LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>17</td>
<td>36</td>
<td>5</td>
</tr>
</tbody>
</table>

### Table: STUDYPDF

<table>
<thead>
<tr>
<th>START</th>
<th>END</th>
<th>LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Pre-Treatment</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Treatment</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1 Week</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2 Weeks</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1 Month</td>
</tr>
</tbody>
</table>
STEP#3: Build the labels associated with the range values established in STEP #2 (macro field LBL#, where # is number 1 through the value stored in macro field: NCOLPDS)

```
data _null_;  
  set studypdf;  
  %do col = 1 %to &ncolpds;  
    %global lbl&col;  
    if(start = &col)then do;  
      call symput("lbl&col",trim(left(label)));  
    end;  
  %end;  
run;
```

SUGGESTION: if you were to package the CNTLOUT statements for STUDYPD and STUDYPDF with STEPS 1-3, all noted above, into a saved macro program, and then called this macro program at the beginning of each program you are submitting, your **standard coding mechanism** would be available during the entire SAS session.

**UTILIZING THE RESULTING MACRO FIELDS AND THEIR VALUES**

All of the building blocks for the **standard coding mechanism** we are trying to achieve are now available within a SAS Session.

For example, use of the following code:

```
%put --There are &ncolpds Analysis Windows --;  
%do col = 1 %to &ncolpds;  
  %put Column &col -- Range: &&beg&col to &&end&col -- &&lbl&col;  
%end;  
```

Results in the following SAS Log entry:

```
--There are 5 Analysis Windows --  
Column 1 -- Range: -5 to -1 -- Pre-Treatment  
Column 2 -- Range: 0 to 0 -- Treatment  
Column 3 -- Range: 1 to 7 -- 1 Week  
Column 4 -- Range: 8 to 16 -- 2 Weeks  
Column 5 -- Range: 17 to 36 -- 1 Month
```

**TESTING WITH ACTUAL DATA**

Let’s create a dataset, WINDOW, with data to show how this approach works:

```
data window;  
  subject=3; day = -4; total = 1; output;  
  subject=3; day = 0; total = 2; output;  
  subject=3; day = 2; total = 3; output;  
  subject=3; day = 9; total = 4; output;  
  subject=3; day = 24; total = 5; output;  
  subject=4; day = -2; total = 6; output;  
  subject=4; day = 0; total = 7; output;  
  subject=4; day = 4; total = 8; output;  
  subject=4; day = 11; total = 9; output;  
  subject=4; day = 29; total = 10; output;  
run;
```
## UTILIZING THE STANDARD CODING MECHANISM IN CALCULATIONS

We are now able to use the **Standard Coding Mechanism** information within a datastep to calculate Analysis Windows.

For example, the following code utilizes the beginning and ending values in macro fields BEG# and END#, respectively, to calculate Analysis Windows stored in the field WINDOW when processing the SAS Dataset WINDOW:

```sas
data window1;
  set window;
  %do col = 1 %to &ncolpds;
    if(&beg&col <= day <= &end&col)then studypd = &col;
  %end;
run;
```

Assuming that this code was part of a Macro Program named WINDOW, we would find the following entry in our log:

```sas
MPRINT(WINDOW):   data window1;
MPRINT(WINDOW):   set window;
MPRINT(WINDOW):   %do col = 1 %to &ncolpds;
MPRINT(WINDOW):       if(&beg&col <= day <= &end&col)then studypd = &col;
MPRINT(WINDOW):   %end;
MPRINT(WINDOW):   run;
```

### Contents of dataset: WINDOW:

<table>
<thead>
<tr>
<th>subject</th>
<th>day</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>-4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>-2</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>29</td>
</tr>
</tbody>
</table>

### Contents of dataset: WINDOW1:

<table>
<thead>
<tr>
<th>subject</th>
<th>day</th>
<th>total</th>
<th>studypd</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>-4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>24</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>29</td>
<td>10</td>
</tr>
</tbody>
</table>
CONVERTING THE DATA TO REFLECT THE TIMEFRAMES

The following code will convert the data so as to array the data by Subject and Timeframe Window:

```sas
proc sort data=window1;
  by subject;
run;

proc transpose data=window1 prefix=col out=window2;
  var total;
  id studypd;
  by subject;
run;
```

Results in the following contents in dataset WINDOW2:

<table>
<thead>
<tr>
<th>Subject</th>
<th>NAME OF FORMER VARIABLE</th>
<th>col1</th>
<th>col2</th>
<th>col3</th>
<th>col4</th>
<th>col5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 local</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>4 local</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

UTILIZING THE STANDARD CODING MECHANISM WITHIN PROC REPORT

We are now able to use this information to generate a table within PROC REPORT.

The following code within a macro named WINDOW utilizes the beginning and ending values and the labels, discussed above, to create a dynamic PROC REPORT statement:

```sas
ods escapechar='^';
ods listing close;
ods rtf body="c:\xxx.rtf";

proc report data=window2 split='\' nowd;
  col subject ('Timeframe Windows' col1-col&ncolpds);
  define subject / display 'Subject';
  %do col = 1 %to &ncolpds;
    define col&col / display '# &col$Days: &&beg&col thru &&end&col$&&lbl&col'
      style={cellwidth=1.2in};
  %end;
run;
ods rtf close;
ods listing;
```

The resultant log entry is as follows:

```
MPRINT(WINDOW):   ods escapechar='^';
MPRINT(WINDOW):   ods listing close;
MPRINT(WINDOW):   ods rtf body="c:\xxx.rtf";
proc report data=window2 split='\' nowd;
  col subject ('Timeframe Windows' col1-col5);
  define subject / display 'Subject';
  %do col = 1 %to &ncolpds;
    define col&col / display #' 1$Days: -5 thru -1$Pre-Treatment'
      style={cellwidth=1.2in};
  %end;
run;
```

```
MPRINT(WINDOW):   ods escapechar='^';
MPRINT(WINDOW):   ods listing close;
MPRINT(WINDOW):   ods rtf body="c:\xxx.rtf";
NOTE: Writing RTF Body file: c:\xxx.rtf
```

```
MPRINT(WINDOW):   proc report data=window2 split='\' nowd;
MPRINT(WINDOW):   col subject ('Timeframe Windows' col1-col5);
MPRINT(WINDOW):   define subject / display 'Subject';
MPRINT(WINDOW):   define col1 / display '# 1$Days: -5 thru -1$Pre-Treatment'
MPRINT(WINDOW):     style={cellwidth=1.2in};
MPRINT(WINDOW):   define col2 / display '# 2$Days: 0 thru 0$Treatment'
MPRINT(WINDOW):     style={cellwidth=1.2in};
MPRINT(WINDOW):   define col3 / display '# 3$Days: 1 thru 7$1 Week'
MPRINT(WINDOW):     style={cellwidth=1.2in};
MPRINT(WINDOW):   define col4 / display '# 4$Days: 8 thru 16$2 Weeks'
MPRINT(WINDOW):     style={cellwidth=1.2in};
MPRINT(WINDOW):   define col5 / display '# 5$Days: 17 thru 36$1 Month'
MPRINT(WINDOW):     style={cellwidth=1.2in};
run;
```
The resultant ODS based .rtf file would look something like the following:

CONCLUSION
In summary, using PROC FORMAT to create macro variables based upon timeframe parameters provides a powerful tool for creating a standard coding mechanism for processing fields, tables and listings based upon Analysis Windows.

Should the Analysis Windows change for any reason, be it additional windows are needed, or the day ranges in the windows need to be revised, the only change necessary is to the program containing the formats directly responsible for the calculation of these timeframes. All other programs can remain as they were provided that they utilized this technique in the calculation of analysis timeframe windows.

REFERENCES:
1 For specifics on using permanent format libraries see the section titled ‘Storing Informats and Formats’ in http://support.sas.com/documentation/cdl/en/proc/59565/HTML/default/a000146279.htm
2 For more information on Global Macro Variables please see http://support.sas.com/documentation/cdl/en/mcrolref/59526/HTML/default/a000206954.htm

ACKNOWLEDGMENTS
I would like to thank my employer W. L. Gore & Associates for allowing me the time to develop and present this paper. In addition, I would like to thank my colleagues and friends Bruce Spotts, Dante diTommaso and Alan Schussman for their assistance in reviewing and providing suggestions for this paper and resultant presentation.

CONTACT INFORMATION
Your comments and questions are valued and encouraged. Please feel free to contact the author at:

Marc Mucatel
W. L. Gore & Associates
4250 W. Killie Lane
Flagstaff, AZ  86001
E-mail: mmucatel@wlgore.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are trademarks of their respective companies.