Building Calendars with Proc Report
Frank Bereznay, IBM

ABSTRACT
Web based calendars are a very intuitive way to organize and present large sets of chronologically related reporting objects. Each day or period of time becomes a link to the related content. Cell attribution, such as background colors (traffic lighting) and or pop up values, can also be used to enhance the information content for a calendar cell. This paper will show how SAS Proc Report can be used to create these types of calendars.

INTRODUCTION
Calendars are a popular object on web pages because they are easy to understand. Generally they are used to select a date, but the same format could be used as a basis for navigating and linking to reporting objects associated with a date. This type of a navigation and data presentation has the ability to present multiple years of reporting object links in a single web page with a very intuitive easy to understand interface.

Initially it was thought that SAS’s Proc Calendar would be an ideal solution to create a calendar to act as a navigation technique to reporting objects associated with a date, but the Calendar procedure did not have the ability to attribute cells in a straightforward way. It turns out Proc Report is well suited to create these types of calendars.

Proc Report is a very powerful and popular reporting tool. One of the best ways to measure the popularity and flexibility of this tool is to see how many user experience papers have been written about it. A scan of SAS’s Global Forum online proceedings index produced the list of papers in the reference section. These papers are a small subset of the total volume of papers on Proc Report usage. Many of these papers expand on the coding techniques used in this paper. In addition to user experience papers, Art Carpenter has written a book on how to use Proc Report which the author has found very helpful and useful.

The initial motivation for developing a calendar like this was to provide a navigation technique to large volumes of chronologically related reporting objects. Calendars serve this requirement well. Decades of historical reporting can be easily and intuitively located from a single page. As we gained experience with the generation of these calendars, we expanded the scope and began to add a status color and then a popup value to the cells. The calendars transitioned from being a navigation aid to becoming a status reporting tool. A fully attributed calendar with color coded cells, drill down reports and popup values has been called a heat chart style dashboard for the subject matter. This paper will share the Proc Report coding techniques developed to create these types of calendars. Two examples will be provided. The first is a simple calendar with no cell status information and the second example will demonstrate how cells can be attributed to provide status information.

CALENDAR CONSTRUCTION – A TWO STEP PROCESS
Creating a Proc Report calendar is a two step process. The first step is to create a calendar dataset with one observation per week for the date range of the calendar. The second step will use Proc Report to create the calendar. Two examples will be provided: (1) A Basic Calendar and (2) A Calendar with ‘traffic lighted’ cells, cell popup displays and summary columns.

EXAMPLE1 – BASIC CALENDAR
Step 1 – Build a Calendar Dataset
The creation of a calendar dataset can be conceptually viewed as transforming a vertical vector of date observations into a set of horizontal date vector observations with one week of date values per observation. A DO loop is typically used to create the vertical vector of date observations. An array and the function WEEKDAY is used to create the weekly horizontal vector observations. Figure 1 shows an example of a SAS data step to build a calendar dataset followed by a Proc Print to display it, which is shown as Figure 2.
An Array, Days, is defined which contains seven elements, corresponding to the days of the week. This array is initialized to missing values and the individual elements are formatted to the Date5. specification (DDMon). A Do loop is used to create the date values, in this case from January 1, 2012 to March 31, 2012. The next replacement statement is the key to the transformation task. The WeekDay function accepts a SAS date value and returns a number from 1 (Sunday) to 7 (Saturday) which corresponds to the day of the week for that particular date. The day of week value is used as an index to place that date value in the proper location in the Days array. For most applications, three calendar months is a good fit for a web page. The variable Quarter is created to act as a By group variable to accomplish this. The next two IF statements determine if it is time to output an observation, either when the week is full (The Saturday array element is populated) or the end of the month is reached. This process is repeated for the date range to complete the calendar dataset. A final If statement is used to output any partial week that may be present when the Do loop is exited.

```sas
Data Calendar;
  Array Days(7) Sun Mon Tue Wed Thu Fri Sat (7*);
  Format Sun Mon Tue Wed Thu Fri Sat Date Date5.;
  Drop Date X;
  Do Date = '01Jan2012'D to '31Mar2012'D by 1;
    Days(WeekDay(Date)) = Date;
    Quarter = Put(Date,YYQ6.);
    If WeekDay(Date) = 7 Then
      Do; Output;
        Do X=1 to Dim(Days); Days(X)=. ; End;
      End;
    If Month(Date) NE Month(Date+1) Then
      Do; Output;
        Do X=1 to Dim(Days); Days(X)=. ; End;
      End;
      End;
    If NMiss(Sun,Mon,Tue,Wed,Thu,Fri,Sat) < 7 Then Output;
  Proc Print;
  Title1 'Calendar Data Step Output';
  Run;
```

Figure 1. Example 1 – Calendar Data Step

Figure 2. Example 1 – Proc Print Output
The calendar dataset has the same structure as the calendar we wish to create, one observation per week. The use of missing values for non date cells is intentional. The missing values will be tested in the Proc Report logic to determine if a particular date cell should be attributed or acted upon. The final row of missing values is an anomaly when the end of a month is also the end of a week. A more sophisticated output routine can be developed to eliminate the spurious observation, but was omitted from this example to keep it simple.

Step 2 – Create Calendar with Proc Report

The Proc Report code to build a calendar is shown as Figure 3. The best way to understand Proc Report execution logic is to start with the Column Statement. This statement will define the variables that will appear in the report and the order in which they will appear. In this example the Column Statement contains the seven day of week variables. The second step is to locate the Define statements for each of the Column Statement variables to see what types of actions are going to be applied to the variables. In this case each variable is being presented as a display variable. The last Proc Report code construct are the Compute Blocks. These 'mini data steps' are used to perform a large variety of tasks to the reporting variables. Compute Blocks which include a variable name in the Compute statement will be executed every time the variable is processed. There is a compute block for each day variable and the code (a Call Define statement) is used to create an URL link for the calendar cell. A cell will be attributed only if it contains a valid date value. Since the same code is repeated for each day variable, a macro (Compute), with one keyword parameter was used. The constructed URL is unique to each cell since it contains the date value as part of the URL. These links will point to the related reporting objects for the day or period of time represented by the cell.

The Proc Report code is ‘sandwiched’ by ODS statements to direct the output to the HTML destination. The Body/Contents/Frame keyword parameter set creates a table of contents and separate page for each By Group page that Proc Report produces. The web page produced by this program is shows as figure 4.

```
Let Path=%Str(C:\Documents and Settings\Administrator\My Documents\);
%Let PathHTML=%Str(&Path.Calendar\HTML\);
Options Missing='';
%Macro Compute(Day=);
  If &Day NE . Then
    Call Define("&Day","URLP",
    Cats("URL_Prefix_",LowCase(Put(&Day,Date.)),".html"));
%Mend Compute;
ODS Listing Close;
ODS HTML Body="cal-body1.html"(Title="Calendar")
    Contents="cal-contents.html"(Title="Calendar")
    Frame="calendar.html"(Title="Calendar")
    Path="&PathHTML"(url=None)
    Headtext="<BASE target=_blank >'"
    Newfile=Page
    Style=SASWeb;
ODS Proclabel "Label for Table of Contents";
Proc Report Data=Calendar Contents="" NoWD;
Column Sun Mon Tue Wed Thu Fri Sat;
  Define Sun / Display;
  Define Mon / Display;
  Define Tue / Display;
  Define Wed / Display;
  Define Thu / Display;
  Define Fri / Display;
  Define Sat / Display;
  Compute Sun; %Compute(Day=Sun) EndComp;
  Compute Mon; %Compute(Day=Mon) EndComp;
  Compute Tue; %Compute(Day=Tue) EndComp;
  Compute Wed; %Compute(Day=Wed) EndComp;
  Compute Thu; %Compute(Day=Thu) EndComp;
  Compute Fri; %Compute(Day=Fri) EndComp;
  Compute Sat; %Compute(Day=Sat) EndComp;
By Quarter;
Title1 "Title for Web Page - Proc Report Calendar";
Run;
ODS HTML Close;
ODS Listing;
Run;
```

Figure 3. Example 1 – Proc Report Code
Figure 4. Example 1 – Proc Report Output

A basic calendar like this first example is primarily a method to navigate to reporting objects associated with a particular day. The related reporting objects can be anything that can be linked to. The calendar itself does not provide any sort of status information. It simply provides a very intuitive way to organize the related reporting objects. In this example a single quarter’s dates are displayed. The table of contents can hold many years worth of quarter tables and it will scroll when/if the number of entries exceeds what can be displayed on a single web page.

EXAMPLE 2 – POPUP VALUES, COLOR CODING AND SUMMARY COLUMNS

Example 2 will introduce three new features: (1) Popup Values, (2) Color Coding for cells and (3) Summary Columns. Popup values are a status box that will appear on the screen when the cursor is hovered over a cell. This is also called a fly over value. Color Coding create what is commonly called traffic lighting for cells. Each cell is colored to reflect a status. Summary columns are an idea that Capacity Analyst Yogesh Shah came up with. During a presentation of this reporting capability at a local CMG user group, Yogesh Shah suggested that the calendar page should be expanded to include weekly, monthly, quarterly and annual reporting links because it was very common to report at those intervals in addition to daily reporting. Thanks to Yogesh Shah for suggesting this, it is now part of the coding structure and be easily added if desired.

Example 2 will report on hourly temperature data at four major US airports, Los Angeles (LAX), Dallas Forth Worth (DFW), Chicago O’hare (ORD), and New York John F Kennedy (JFK) for 2011. Reporting will be done at the daily, weekly, monthly, quarterly and annual intervals. The calendar cell will be color coded to reflect the high temperature for the interval being reported on. These colors will range from blue for 0 degrees, to green for 72 degrees to red at 100 degrees. A popup box will be constructed for each cell showing the low, average and high temperature for the cell interval. Each cell will link to a line plot and tabular data page for the temperature data.

Step 1A – Build Vertical Vector of Dates – Input SAS Dataset for the Calendar Data Step

Like the previous example, we will begin with a vertical vector of date values, in this example from January 1, 2011 to December 31, 2011. In addition to the date variable, we will need two additional data values per date or interval: (1) a 36 Byte Popup character string, and (2) a Color value for the cell. The Popup character string will contain the Low, Average and High temperature for the date or time interval.
This example is also providing four summary columns: (1) Weekly, (2) Monthly, (3) Quarterly and (4) Annual. These columns will appear to the right of the day columns and will have the same information for the time interval being reported, so each summary interval will also need the 3 variables described above. The vertical vector has become a matrix with 5 reporting intervals (Daily, Weekly, Monthly, Quarterly and Annual) * 3 variables or 15 columns (17 columns when you include the By Group variable, Quarter and a location variable, Loc, to identify the airport) and 365 * 4 or 1,460 observations, one for each day of the year for each airport. The columns associated with the summary intervals will be sparsely populated. For example the monthly variables will have missing values except for the last day of the month when the variables will contain monthly summary variables. A Proc Contents for this SAS data set is shown as Figure 9. This data set will replace the Do Loop used to create a date range to build a Calendar data set for the airport temperature calendar. The SAS code to import the airport temperature data and create this SAS data set is included as Appendix A.

### Proc Contents of Temperature Calendar

**Vertical Matrix of Date and Attribute Variables**

**One Observation per day per Airport – 1,460 Observations**

This Data Set will be the Input to the Calendar Data Step

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Format</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Date</td>
<td>Num</td>
<td>8</td>
<td>DATE.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Dly_Color</td>
<td>Char</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Dly_Popup</td>
<td>Char</td>
<td>36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Loc</td>
<td>Char</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Mth_Color</td>
<td>Char</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Mth_End_Date</td>
<td>Num</td>
<td>8</td>
<td>DATE.</td>
<td>Month</td>
</tr>
<tr>
<td>8</td>
<td>Mth_Popup</td>
<td>Char</td>
<td>36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Qtr_Color</td>
<td>Char</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Qtr_End_Date</td>
<td>Num</td>
<td>8</td>
<td>DATE.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Qtr_Popup</td>
<td>Char</td>
<td>36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Quarter</td>
<td>Num</td>
<td>8</td>
<td>YYQ4.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Wek_Color</td>
<td>Char</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Wek_End_Date</td>
<td>Num</td>
<td>8</td>
<td>DATE.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Wek_Popup</td>
<td>Char</td>
<td>36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Yr_Color</td>
<td>Char</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Yr_End_Date</td>
<td>Num</td>
<td>8</td>
<td>DATE.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Yr_Popup</td>
<td>Char</td>
<td>36</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5. Example 2 – Proc Contents Output**

### Step 1B – Calendar Data Step

Four capabilities have been added to the calendar data step: (1) Cells are color coded based on temperature readings, (2) A popup value has been added as a cell attribute, (3) Four summary columns have been added, and (4) The program has been placed inside a Macro to simplify running it for the four airports. This will create four separate Proc Report input data sets. In addition, the program was changed to read a data set as input in place of the Do Loop used in the previous example.

Two additional arrays, Color and Popup, has been defined. All arrays have been expanded from seven elements to eleven elements to accommodate the summary columns. A Set statement has replaced the Do Loop as a method to obtain the vertical set of date values. The same replacement logic is used as before, the WeekDay function is used to properly insert the date value in the arrays. A slightly different technique is used to populate the summary columns and a Macro, Assign_Values, was developed to simplify coding. The spc variable is created to create a small break between the week day columns and the summary columns. This code is shown as Figure 6.

The output from an execution of this data step will produce the same set of weekly observations as did the other examples. However, this time there will be more columns, 11 vs 7, and there will be three variables for each cell, a date, a background color and a popup character string.
Step 2 – Create Calendar with Proc Report

We start with the Column statement again. It has now grown to 34 variables. This represents 11 reporting columns with 3 variables per column plus the space (spc) column to provide a visual break. Each of the reporting columns has a date variable preceded by a color and popup variable. The Define statements for the popup and color variables use a NoPrint option. This will make the data available for cell attribution but not display it as a column. The Define statements for the summary columns include a column label because, unlike the day variables, the variable names are not good column labels.

The Compute macro now contains four keyword parameters. The three new parameters are popup, color and format. Popup and color provide the popup character string and cell color. The format parameter was added because we wanted to format the presentation of the date value differently for the summary columns. This will permit all the reporting objects to be in the same directory and not have any naming conflicts. The attribution of a popup value is done as a second style element in the Call Define statement that also provides the cell background color. The Proc Report SAS code is shown as Figure 7.

The Proc Report Macro calls are ‘sandwiched’ between ODS code for the HTML destination. This technique places all four airports in a single Table of Contents and each airport entry will have four quarterly tables. The calendar created by this program is shown as figure 8.

SUMMARY

This paper has provided a coding technique to create calendars with Proc Report. We started with a basic calendar to provide navigation to a collection of reporting objects and proceeded to a fully attributed calendar that can also act as a ‘heat chart’ dashboard for the subject. This reporting format has been very useful and it is hoped that others can make use of it too.

If the Calendar program is also responsible for the related drill down reporting objects, as is the case with the temperature calendar, then some additional coding is necessary to produce the needed plots and tables.

The author is happy to share this code. Please send an email to Frank@Bereznay.Info for a copy of the SAS programs, including the generation of the related reporting objects, and supporting airport temperature data.

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.

REFERENCES

%Macro Assign_Values(Int=,X=);
  If &Int._End_Date NE . Then
  Do;
     Days(&X.) = &Int._End_Date;
     Color(&X.) = &Int._Color;
     Popup(&X.) = &Int._Popup;
  End;
%Mend Assign_Values;

%Macro Build_Calendar_Dataset(LOC=);
  Data Cal_&LOC;
  Array Color(11)$ CSun CMon CTue CWed CThu CFri CSat CMth CQtr Cyr (11' ');
  Array Popup(11)$ PSun PMon PTue PWed PThu PFri PSat PWek PMth PQtr Pyr (11' ');
  Array Days(11) Sun Mon Tue Wed Thu Fri Sat Wek Mth Qtr Yr (11* ');
  Format Sun Mon Tue Wed Thu Fri Sat Wek Mth Qtr Yr Date5. Yr Year. Wek MMDDYYD. Mth MONYY.
  Quarter Qtr YYQ.;
  Keep Sun Mon Tue Wed Thu Fri Sat Wek Mth Qtr Yr
     CSun CMon CTue CWed CThu CFri CSat CWek CMth CQtr Cyr
     PSun PMon PTue PWed PThu PFri PSat PWek PMth PQtr Pyr
     Loc Quarter Spc;
  Set Cal_Data_Step_Input End=EOF;
  Where LOC = "&LOC";
  %Assign_Values(Int=Yr,X=11)
  %Assign_Values(Int=Qtr,X=10)
  %Assign_Values(Int=Mth,X=9)
  %Assign_Values(Int=Wek,X=8)
  Days (WeekDay(Date)) = Date;
  Color(WeekDay(Date)) = Dly_Color;
  Popup(WeekDay(Date)) = Dly_Popup;
  Spc = '';
  If WeekDay(Date) = 7 Then
    Do;
    Output;
    Do X=1 to Dim(Days); Days(X)=.; Color(X)=''; Popup(X)=''; End;
    End;
  If Month(Date+1) NE Month(Date) Then
    Do;
    Output;
    Do X=1 to Dim(Days); Days(X)=.; Color(X)=''; Popup(X)=''; End;
    End;
  IF EOF Then
    Do;
    If NMiss(Sun,Mon,Tue,Wed,Thu,Fri,Sat) < 7 Then Output;
    End;
  Run;
%Mend Build_Calendar_Dataset;

%Build_Calendar_Dataset(LOC=DFW)
%Build_Calendar_Dataset(LOC=LAX)
%Build_Calendar_Dataset(LOC=JFK)
%Build_Calendar_Dataset(LOC=ORD)

Figure 6. Example 2 – Calendar Data Step
Figure 7. Example 2 – Proc Report Code

```sas
Options Missing = ' '; %Macro Compute(Day=,Color=White,Format=Date.,Popup=); If &Day NE . Then Do; Call Define("&Day","URLP", LowCase(Cats("&LOC",Put(&Day,&Format),".html"))); Call Define("&Day","style", Cat("style={background=",&Color,"","flyover=",&Popup,"}")); End; %Mend Compute; %Macro Calendar(LOC=); ODS Proclabel "Temperature Calendar for &LOC"; Proc Report Data=Cal.Cal_&LOC Contents="Link" NoWD; Column CSun PSun Sun CMon Mon Mon CTue Tue Tue CWed Wed Wed CFri Fri Fri CSat PSat Sat Spc CWek PWek Wek CMth PMth Mth CQtr PQtr Qtr Cyr Yr; Define CSun / NoPrint; Define PSun / NoPrint; Define Sun / Display; Define CMon / NoPrint; Define PMon / NoPrint; Define Mon / Display; Define CTue / NoPrint; Define PTue / NoPrint; Define Tue / Display; Define CWed / NoPrint; Define PWed / NoPrint; Define Wed / Display; Define CFri / NoPrint; Define PFri / NoPrint; Define Fri / Display; Define CSat / NoPrint; Define PSat / NoPrint; Define Sat / Display; Define Spc / " " Display; Define CWek / NoPrint; Define PWek / NoPrint; Define Wek / 'Weekly' Display; Define CMth / NoPrint; Define PMth / NoPrint; Define Mth / 'Monthly' Display; Define CQtr / NoPrint; Define PQtr / NoPrint; Define Qtr / 'Quarterly' Display; Define CYr / NoPrint; Define PYr / NoPrint; Define Yr / 'Annual' Display; Compute Sun; %Compute(Day=Sun,Color=CSun,Popup=PSun) EndComp; Compute Mon; %Compute(Day=Mon,Color=CMon,Popup=PMon) EndComp; Compute Tue; %Compute(Day=Tue,Color=CTue,Popup=PTue) EndComp; Compute Wed; %Compute(Day=Wed,Color=CWed,Popup=PWed) EndComp; Compute Thu; %Compute(Day=Thu,Color=CThu,Popup=PThu) EndComp; Compute Fri; %Compute(Day=Fri,Color=CFri,Popup=PFri) EndComp; Compute Sat; %Compute(Day=Sat,Color=CSat,Popup=PSat) EndComp; Compute Wek; %Compute(Day=Wek,Color=CWek,Popup=PWek, Format=Julian.) EndComp; Compute Mth; %Compute(Day=Mth,Color=CMth,Popup=PMth, Format=MONYY.) EndComp; Compute Qtr; %Compute(Day=Qtr,Color=CQtr,Popup=PQtr, Format=YYQ6.) EndComp; Compute Yr; %Compute(Day=Yr,Color=CYr,Popup=PYr, Format=Year.) EndComp; By Quarter; Title1 "Temperature Calendar for &LOC"; Run; %Mend Calendar; ODS Listing Close; ODS HTML Body="Cal_Body0.html"(Title="Calendar") Contents="calendar_contents.html"(Title="Calendar") Frame="airport_temperature_calendar.html"(Title="Calendar") Path="/pathHTML"(url=none) Headtext='<BASE target=_blank >' Newfile=Page Style=SASWeb; %Calendar(LOC=LAX) %Calendar(LOC=DFW) %Calendar(LOC=ORD) %Calendar(LOC=JFK) ODS HTML Close;
```
Figure 8. Example 2 – Proc Report Output
APPENDIX A – SAS CODE TO CREATE CALENDAR DATA STEP INPUT FROM AIRPORT TEMPERATURE CSV FILES

LibName LIBRARY "&Path\Calendar\";
LibName CAL "&Path\Calendar\";
%Let Program=%Str(Proc Report Macro);
Options Missing = '';
ODS HTML Close;
ODS Listing;
%Macro Import_Temp(Loc=);
  FileName Air_Temp "&Path\Calendar\&Loc..csv";
  Data &Loc;
    Infile Air_Temp Length=Rec_Len FirstObs=2;
    Length Rec_Check $10 Loc $3.;
    Keep Loc Date Hour Temp;
    Format Date Date.;
    Input  Rec_Check $1-10  Loc $;
    Input @01 Rec_Buffer $Varying100. Rec_Len;
    Date = Input(Scan(Rec_Buffer,1," ",MMDDYY10.));
    Time = Input(Scan(Rec_Buffer,2," ",Time.));
    Temp = Input(Scan(Rec_Buffer,2," ",5.));
    Hour = Hour(Time);
    Loc  = "&Loc";
  Run;
%Mend Import_Temp;
%Import_Temp(Loc=DFW)
%Import_Temp(Loc=JFK)
%Import_Temp(Loc=LAX)
%Import_Temp(Loc=ORD)
Data Temperatures;
  /* This will be the SAS Table used to create temperature plots */
  /* and provide the input to Proc Summary */
  Set DFW JFK LAX ORD;
  Dom = Day(Date);
  Wek = Intnx('Week',Date,0,'E');
  Mth = Intnx('Month',Date,0,'E');
  Qtr = Intnx('Qtr' ,Date,0,'E');
  Yr  = Intnx('Year',Date,0,'E');
  Proc Sort Data=Temperatures Out=Cal.Temperatures;
  By Loc Date;
  TITLE1 "Contents of Temperature Table";
  Proc Datasets Lib=Cal;
    Modify Temperatures;
    Format Temp 5.1;
    Format Date Wek Mth Qtr Yr Date. ;
    Label Temp = 'Temperature'
      Mth = 'Month'
      Dom = 'Day of the Month';
    Index Create Loc_Dat=(Loc Date);
    Index Create Loc_Wek=(Loc Wek);
    Index Create Loc_Mth=(Loc Mth);
    Index Create Loc_Qtr=(Loc Qtr);
    Index Create Loc_Yr=(Loc Yr);
  Contents Data=Cal.Temperatures;

Continued on next page
Proc Summary Data=CAL.Temperatures;
  Class Loc Date Wek Mth Qtr Yr;
  Var Temp;
  Output Out=Summary_Temp
    Min=Low Mean=Average Max=High;
Data Annual (Rename=(Low=Yr_Low Average=Yr_Avg High=Yr_High
    Yr=Yr_End_Date));
  Quarter(Rename=(Low=Qtr_Low Average=Qtr_Avg High=Qtr_High
    Qtr=Qtr_End_Date));
  Monthly(Rename=(Low=Mth_Low Average=Mth_Avg High=Mth_High
    Mth=Mth_End_Date));
  Weekly (Rename=(Low=Wek_Low Average=Wek_Avg High=Wek_High
    Wek=Wek_End_Date));
  Daily  (Rename=(Low=Dly_Low Average=Dly_Avg High=Dly_High));
  Drop_Type_ _Freq_;
Set Summary_Temp;
Select(_Type_);
  When(33) Do; Date = Yr; Output Annual; End;
  When(34) Do; Date = Qtr; Output Quarter; End;
  When(36) Do; Date = Mth; Output Monthly; End;
  When(40) Do; Date = Wek; Output Weekly; End;
  When(48) Output Daily;
  Otherwise;
End;
%Macro Attribute(Dat=,Int=);
  If &Dat. NE . Then 
    Do;
      &Int._Color = Put( &Int._High,Color.);
      &Int._Popup = Cat(''','Put( &Int._High,3.),' - High','0D'x,
        Put( &Int._Avg ,3.),' - Average','0D'x,
        Put( &Int._Low ,3.),' - Low','''');
    End;
%Mend Attribute;
Data Cal.Cal_Data_Step_Input;
  /* This will be the SAS table used to create the Calendars */
  Length Dly_Color Wek_Color Mth_Color Qtr_Color Yr_Color $8.;
  Dly_Popup Wek_Popup Mth_Popup Qtr_Popup Yr_Popup $36.;
  Keep Loc Quarter Date Dly_Color Wek_End_Date Wek_Color
    Wek_Popup Mth_End_Date Mth_Color Mth_Popup Qtr_End_Date Qtr_Color
    Qtr_Popup Yr_End_Date Yr_Color Yr_Popup;
  Format Quarter YY4.;
  Merge Annual Quarter Monthly Weekly Daily;
  By Loc Date;
  %Attribute(Dat=Date,Int=Dly)
  %Attribute(Dat=Wek_End_Date,Int=Wek)
  %Attribute(Dat=Mth_End_Date,Int=Mth)
  %Attribute(Dat=Qtr_End_Date,Int=Qtr)
  %Attribute(Dat=Yr_End_Date,Int=Yr)
Title1 "Contents of Calendar Data Step Input Table";
Proc DataSets Lib=Cal;
  Modify Cal.Cal_Data_Step_Input
    (Label='Output from CSV Import / Input to Build Calendar Table');
  Contents Data=Cal.Cal_Cal_Data_Step_Input;
Quit;