

Report Destination Macro

Mark Terjeson
Russell, Tacoma, WA

ABSTRACT

The need was to be able to create Excel or CSV output easily from SAS or SAS web server and eliminate the labor time in customizing output for each SAS application built but yet have professional looking reports.

INTRODUCTION

While SAS version 9 or newer versions have a greater number of features and capabilities, this project originated during the use of SAS version 8.2 and is still used today in both v8.2 and v9 environments.

Two straightforward elements led to the sophisticated tool represented here. Once SAS and ODS were utilized for SAS output and SAS web server output, the initial tweaking of the PROC TEMPLATE tool for ODS yielded reasonably professional looking reports for the application report building efforts. The adjustments to the PROC TEMPLATE coding syntaxes was not without challenges to locate documentation in the early days of its use. Other challenges for the less experienced made customizing ODS to Excel output a laborious effort. Web investigation and documentation hunting eventually led to acquiring leads to most of the PROC TEMPLATE attribute capabilities. Of course once you get a routine established to create the type and style of report output you are looking for you then have a working template or example to give you a boost for the next different report that you want to build. After the initial development efforts and learning curves have been expended you now are in a position of straightforward replication of your ODS PROC TEMPLATE outputs. The second straightforward element to this project is mere substitution of parameters in a routine which software programming is all about. The great SAS Macro tools make text and parameter substitution a snap. The macro tools and techniques used may seem to be advanced but are nothing more than an accumulation of many simple parameter text substitutions, one intermediate technique of dynamic argument acquisition, and some caveat behavior accommodation.

IN THE BEGINNING

Like any good programmer, coding samples from SAS manuals, books, online web sites, and user lists such as SAS-L were gathered to learn how to customize the ODS PROC TEMPLATE to get the basic cosmetic adjustments to the Excel output that were desired. The PROC TEMPLATE provides the ability to place custom coding in a variety of event triggered locations. The DOC_HEAD event is used to embed some global workbook settings. The TABLE event is used to establish overall row heights, column widths and justifications. The DATA event is used to control cell attributes and any overriding justifications.

To customize the output there are different reserved words and statements to control or add the desired resultant cosmetic output adjustments within the PROC TEMPLATE define event sections. The coding at this point is good for one desired output layout. If you wish to utilize this output control capability and alter the layout and cosmetics for a different application report output we then merely need to do text substitution to allow us to tweak a single cosmetic element for a dynamic value that we want to pass to it. For example if we were to set the width of a column to be a certain number for this report we may wish to have a different column width for the next report. If we could just pass this value and have the SAS Macro substitute this value in the right place then we have created a dynamic tool. In the next application report output there were one or two other attributes that were desired to be controlled. After building thirty or forty reports the result was accumulating the twenty-five or thirty most used attributes that would be handy to make adjustable with dynamic values. For each of these single parameter substitutions, the macro code was modified for each, and eventually the report destination macro started looking like a sophisticated tool.

ORGANIZATION OF THE MACRO MODULE

The enclosed macro module is a text file containing the report destination macro being described herein and basically has only the three following sections:

```
header-documentation
support macro routines
main macro
```

The header-documentation section has typical title, date, and purpose notations followed by a legend describing briefly each adjustable parameter and its purpose. These are the parameters that the user can dynamically adjust to customize the report output easily in one macro call while the report destination macro does all the hard work of organizing where they all go and how to implement each argument. Following the header-documentation are nine support macro routines that make life easier in a few locations of the main macro. The main macro is the engine that performs all of the desired adjustments and creates the end result.

```
header-documentation
support macros routines
main macro
    dynamic argument acquisition
    establish parameter values
        set default values
        compute derived values
    CSV or EXCEL output
```

The main macro is divided into three sections. The first section (dynamic argument acquisition) is the intermediate/advanced level of macro coding difficulty mentioned earlier. The default methods of passing arguments to a SAS Macro fall into two flavors, positional and non-positional. Either all the argument positions must be provided (even if empty) and represented by all the comma delimiters needed, or the non-positional syntax allows one or more arguments to be provided, regardless of order or position, and the rest can be missing. However, the tricky part is that we take manual control over the arguments passed so that we can parse them ourselves so that a couple of important things are taken care of. We not only need to parse out a few tricky types of arguments but most importantly is that we need to know which were passed and which were not. This allows us to know which were missing and set default values as needed for all the parameters so that nothing will end up incomplete and result in errors from a failing macro routine.

The second section (establish parameter values) is just bulky repetition setting many default values. Also, based upon several of our parameters which we have designed to allow the user to specify dynamic numbers of them, we automatically create other derived variables and values to fill out necessary suites of parameters.

The third section (CSV or EXCEL output) is the main engine that takes all of the dozens of parameters established above and substitutes them into the appropriate locations of the PROC TEMPLATE logic which results in a very dynamic yet powerfully adjustable single call routine that creates a wide variety of report outputs customized to fit each application's needs and unique data layouts and specifications.

CALLING THE MACRO

The report destination macro can be called as minimally as:

```
%rptdest;
```

It can't get much simpler than that. This provides all of the execution of code necessary to create an ODS EXCEL workbook output from the most recent dataset used.

All other arguments supplied to this macro just add the selected cosmetic adjustments or workbook alterations you tell it to do. Any one or more parameters can be added to the list of arguments. For example to add a title and left justify the title you can add the arguments below:

```

%rptdest(
    title=Flintstones,
    titlejustification=l
);

```

If you wanted to add two lines of leading between the title and the data, spread the column widths a little, and override mixed Excel alpha and numeric split justification rules you could add these parameters:

```

%rptdest(
    title=Flintstones,
    titlejustification=l,
    leadingrows=2,
    columnwidth1=10,
    columnwidth2=14,
    columnwidth3=8,
    columnjustification4=r
);

```

This macro was designed so that any parameter token designators such as COLUMNWIDTH# is followed by a number to denote the actual dynamically selectable item such as which column. Example, for all of the parameter names that begin with COLUMN the number the user can specify on the end is the column number to apply the desired parameter to. In the sample code above COLUMN #1 width is set to 10 and COLUMN #2 width is set to 14. In COLUMN #4 the justification is set to "r" which is our macro's shortcut for "Right" justify. The documentation in the header section of the module is briefly descriptive of what types of values the argument is expecting.

Similarly if parameter names begin with TITLE then the suffix number represents the title number. For more adjustments to the title in the report we can bump up the font size, the row height, and tell the routine to do macro-tuning were to position the title in relationship to our data with the TITLECOLUMN# parameter. This places the title into the specified column on the worksheet, to get it into the vicinity you desire. To micro-tune the exact placement of the title, for example to center it over the data, you can specify any number of leading or trailing spaces to the title in order to micro-adjust the lateral positioning.

```

%rptdest(
    title=Flintstones,
    titlejustification=l,
    titlecolumn1=2,
    titlefontsize=18,
    titlerowheight=23.25,
    titleleadingspacecount=2,
    leadingrows=2,
    columnwidth1=10,
    columnwidth2=14,
    columnwidth3=8,
    columnjustification4=r
);

```

Other examples of parameters that have been built into the report destination macro are different destinations, such as CSV, Excel Workbook, and the SAS log.

```

%rptdest(
    data=mydataset,
    destination=csv,
    xlwinpath=c:\temp,
    xlfilename=MyFile,
    sheetname=My Sheet Name,
    debug=131
);

```

EXCEL PARAMETERS CONTROLLED

Other workbook controls implemented in this macro are header and page margin specifications for top, bottom, left and right values. Split character desination can be passed along to SAS PROCs used for display output.

USER CONTROLLED OUTPUT

THE MACRO CODE

```

*****
*
* PROGRAM: rptdest (macro) Version 1.0
*
* WRITTEN: 05/02/2005 by Mark Terjeson
*
* PURPOSE: Automatic report destination macro for CSV or Print Ready reports.
* Useful for single table reports. Automatically prepares PROC TEMPLATE
* with user provided Excel parameters as seen below. The available
* parameters are not required to be passed. If not passed they will
* default to predetermined default values thus saving you having to pass
* them. You may override the default values by passing the nonpositional
* macro arguments that need a custom value. Any of the parameters names
* listed below to the left of the equals sign may be passed separated by
* commas and your new value substituted to the right of the equals sign.
*
*
* optional parameters (original default values shown):
*
* data=&syslast, defaults to last dataset written
*
*
* rundate=%sysfunc(today(),date9.), defaults to todays date
*
*
* destination=prntrdy, case insensitive, csv,
* (anything else defaults to prntrdy)
*
*****

```

```

*
*
*      xlwinpath=,          path for xlfilename (WINDOWS platforms)
*
*
*      xlfilename=rptdest,  Excel Workbook name less extension
*
*
*      sheetname=Sheet1,    Excel Worksheet name (max len 31)
*
*
*      orientation=portrait, case insensitive, landscape,
*                          (anything else defaults to portrait)
*
*
*      fittopage=,          nonblank will trigger fittopage mode
*                          Excel/File/Page Setup/Page/Scaling
*
*
*      fitwidth=1,          Excel/File/Page Setup/Page/Scaling
*
*
*      fitheight=0,         Excel/File/Page Setup/Page/Scaling
*
*
*      rowstorepeat=1,      Excel/File/Page Setup/Sheet/Rows Repeat
*                          | default automatically computed from the
*                          +-----> number of title and leading rows plus 1
*                                  for the column labels
*
*
*      headermargin=0.5,    Excel/File/Page Setup/Margins/Header
*
*
*      topmargin=1.0,        Excel/File/Page Setup/Margins/Top
*
*
*      rightmargin=0.75,     Excel/File/Page Setup/Margins/Right
*
*
*      bottommargin=1.0,     Excel/File/Page Setup/Margins/Bottom
*
*
*      leftmargin=0.75,      Excel/File/Page Setup/Margins/Left
*
*
*      footermargin=0.5,     Excel/File/Page Setup/Margins/Footer
*
*
*      title1=,              title(s) will be placed into the first
*                          rows of worksheet and frozen so they
*                          will be repeated on each page.  if
*                          leading or trailing spaces required to
*                          adjust centering then see parameters
*                          title(lead/trail)ingspacecount below
*
*
*      titlecolumn1=1,        to macro adjust centering of titles
*                          specify the column in which to place
*                          the title text.  to micro adjust the
*                          centering pad title text with trailing
*                          or leading spaces with the parameters
*                          title(lead/trail)ingspacecount below
*
*
*      titlefontsize=12,      font size used for all titles
*
*
*      titlerowheight=15.75, row height for all titles.  must be
*                          adjusted to accommodate changes in the
*                          titlefontsize used
*

```

```

*
*
*      titleleadingspacecount=0,  number of leading spaces to add to
*                                all the titles for micro centering
*
*
*      titletrailingspacecount=0, number of trailing spaces to add to
*                                all the titles for micro centering
*
*
*      titlejustification1=c,    autofills missing titlejustification(N)
*                                to previous titlejustification1(N-1)
*
*
*      leadingrows=0,           number of rows of leading between the
*                                titles and columnlabel row.  i.e. macro
*                                adjustment for leading
*
*
*      leadinggrowheight=12.75, micro adjustment available for leading
*
*
*      columns=1,               number of columns
*
*
*      columnwidth1=8.43,       will autofill missing columnwidth(N)
*                                to previous columnwidth(N-1)
*
*
*      columnjustification1=,    will not autofill missings.  if present
*                                will justify with values l,r,c
*                                thus allows Excel default justification
*                                rules if nothing specified and overrides
*                                Excel rules when present
*
*
*      footer=,                 text positioned in the center block
*                                of the Excel footer
*
*
*      splitcharacter=%str( ),  proc parameter where to split labels
*
*
*      outputmacro=,            blank defaults to internal proc print
*                                nonblank will then use the macro name
*                                supplied that contains the outputting
*                                code such as proc report, etc.
*                                applies to printready only, not csv
*
*
*      columnnumbers=           add a row with column numbers for
*                                development purposes to assist in
*                                indexing suffixes in other parameters.
*                                nonblank will trigger column numbers
*
*
*      debug=                    nonblank will trigger debug code
*                                automatically turned on with _debug use
*
*
*****
*
*  WARNINGS:
*
*      Currently ampersands not allowed in text of the following parameters:
*      titleN or footer text, workbook or worksheet names.
*
*****
*
*  NOTES:
*
*      Parameters with suffix index may have any user defined count.

```

```

*
*      number of columns = max(columns, max(N) of columnwidthN)
*
*      number of titles = max(max(N) of title, max(N) of titlecolumnN)
*
*****
*
* HINTS:
*
*      SETTING COLUMN WIDTHS
*
*      Typically, a quick method of populating columnwidths is to open a mock
*      up of the spreadsheet and adjust to desired widths. Then open up the
*      rptdest columnwidth= code side-by-side and rightclick the spreadsheet
*      column header, open Column Width, control_C(copy the highlighted value),
*      hit cancel, and paste value directly into the columnwidth= location.
*
*
*      Additional Hint: (if header row heights double on you)
*
*      Usually, pasting the column width stays just fine. However, if using
*      a splitcharacter= AND there are column headers that will get wrapped on
*      the specified character, it seems that the column width is automatically
*      adjusted again by Excel after the wrapping on word breaks is performed.
*      The tendency of this after-wrap-adjusting sets the final column width
*      smaller than what you specified. A quick test of running the program
*      and checking the spreadsheet column width results, you can find the one
*      or more columns that are triggering the row height doubling. For these
*      column widths that Excel adjusted-down during the wrapping, merely bump
*      your specified columnwidth= up a little by trial and error until you
*      compensate for the wrap effect.
*
*****
*
* MODIFIED: __/__/___ <your name here> [000]
*          <description here>
*
*****

```

```

*****
*
*      Dependency macros (i.e. nonstandalone)
*
*      for internal use only
*
*****

```

```

%* if missing, declare and set to default ;
%macro _setdefault(mvar,mdefault);

%let __mvarfound=0;
%do __k = 1 %to &ArgCount ;
%if %upcase(%scan(&&_arg&__k,1,'='))
eq %upcase(&mvar) %then %let __mvarfound=1;
%end;
%if &__mvarfound eq 0 %then
%do;
%global &mvar;
%let &mvar=&mdefault;
%end;

%* necessary override for titles to be output. ;
%* must use nested IF to avoid vartype errors. ;
%if &mvar eq columns %then
%if &&mvar lt 1 %then %let columns=1;

%mend;

```

```

    %* validate justification values l,c,r ;
%macro _trimlcr(mvalue);
    %if &mvalue ne %then
        %do;
            %let _lcr=%lowercase(%substr(&mvalue,1,1));
            %if &_lcr eq l
                or &_lcr eq c
                or &_lcr eq r %then %str(&_lcr);
                %else %str();
            %end;
        %end;
    %mend;

    %* establish variables ;
%macro _makevars(mcountvar,mvar);

    %* mcountvar = max(mcountvar, largest_mvar_suffix) ;
    %let _mvarfound=0;
    %let _maxmvaridx=0;
    %do _i = 1 %to &ArgCount;
        %if %upcase(%scan(&&_arg&_i,1,'=')) eq %upcase(&mcountvar) %then %let
            _mvarfound=1;
        %if %sysfunc(compress(%upcase(%scan(&&_arg&_i,1,'=')),1234567890)) eq
            %upcase(&mvar) %then
            %do;
                %let
                __tmpidx=%sysfunc(compress(%upcase(%scan(&&_arg&_i,1,'=')),%upcase(&mvar)));
                %if &__tmpidx gt &_maxmvaridx %then %let _maxmvaridx=&__tmpidx;
            %end;
        %end;
        %if &_mvarfound eq 0 %then %global &mcountvar;
        %if &_maxmvaridx gt &&mcountvar %then %let &mcountvar=&_maxmvaridx;
        %do _i = 1 %to &&mcountvar;
            %global &mvar&_i;
            %if %index(%upcase(&mvar&_i),JUSTIFICATION) ne 0 %then
                %do;
                    %let &mvar&_i=%_trimlcr(&&mvar&_i);
                %end;
            %end;
        %end;
    %mend;

    %* establish max counter from indexes ;
%macro _maxidx(mcountvar,mvar);

    %* mcountvar = max(mcountvar, largest_mvar_suffix) ;
    %do _i = 1 %to &ArgCount;
        %if %sysfunc(compress(%upcase(%scan(&&_arg&_i,1,'=')),1234567890)) eq
            %upcase(&mvar) %then
            %do;
                %let
                __tmpidx=%sysfunc(compress(%upcase(%scan(&&_arg&_i,1,'=')),%upcase(&mvar)));
                %if &__tmpidx gt &&mcountvar %then %let &mcountvar=&__tmpidx;
            %end;
        %end;
    %end;
%mend;

    %* establish max counter from references ;
%macro _maxref(mcountvar,mvar);

    %* counter = max(counter, largest_mvar_value) ;
    %do _i = 1 %to &ArgCount;

```

```

        %if %sysfunc(compress(%upcase(%scan(&&_arg&_i,1,'')),1234567890)) eq
%upcase(&mvar) %then
        %do;
            %let __tmpidx=%scan(&&_arg&_i,2,'');
            %if &__tmpidx gt &&mcountvar %then %let &mcountvar=&__tmpidx;
        %end;
    %end;

%mend;

    %* establish variables and autofill missings ;
%macro _autofill(mcountvar,mvar);

    %* mcountvar = max(mcountvar,largest_mvar_suffix) ;
    %_makevars(&mcountvar,&mvar);

    %* autofill any missing mvars ;
    %let _lastvalue=&&mvar.1;
    %do _i = 1 %to &&mcountvar;
        %let _found=0;
        %do _j = 1 %to &ArgCount;
            %if %upcase(%scan(&&_arg&_j,1,''))
                eq %upcase(&mvar&_i) %then %let _found=1;
        %end;
        %if &_found eq 0 %then
            %do;
                %global &mvar&_i;
                %let &mvar&_i=&_lastvalue;
            %end;
        %let _lastvalue=&&mvar&_i;
    %end;

%mend;

    %* build string of spaces ;
%macro _spaces(num);
    %if &num ne %then
        %do;
            %let theString=;
            %do i = 1 %to &num;
                %let theString=&theString%str( );
            %end;
            %str(&theString)
        %end;
    %end;

%mend;

    %* apply leading/trailing spaces to titles if specified ;
%macro _padtitles;
    %do _idx = 1 %to &titles;
        %if %quote(&&title&_idx) ne %then
            %do;
                %let
title&_idx=%_spaces(&titleleadingspacecount)%quote(&&title&_idx)%_spaces(&titletrailingsp
acecount);
            %end;
        %end;
    %end;

%mend;

    %* display values for debugging purposes ;
%macro _shwvalues(Args) / pbuff;
    %put RPTDEST macro arguments <beg>: &syspbuff;
    %put data is >&data<;
    %put rundate is >&rundate<;

```

```

        %put      destination      is >&destination<;
        %put      xlfilename       is >&xlfilename<;
        %put      sheetname        is >&sheetname<;
        %put      orientation      is >&orientation<;
        %put      fittopage        is >&fittopage<;
        %put      fitwidth         is >&fitwidth<;
        %put      fitheight        is >&fitheight<;
        %put      rowstorepeat     is >&rowstorepeat<;
        %put      headermargin     is >&headermargin<;
        %put      topmargin        is >&topmargin<;
        %put      rightmargin      is >&rightmargin<;
        %put      bottommargin     is >&bottommargin<;
        %put      leftmargin       is >&leftmargin<;
        %put      footermargin     is >&footermargin<;
        %put      titles           is >&titles<;
%do _cc = 1 %to &titles;
        %put      title&_cc       is >%quote(&&title&_cc)<;
        %put      titlecolumn&_cc is >&&titlecolumn&_cc<;
        %put      titlejustification&_cc is >&&titlejustification&_cc<;
%end;
        %put      titlefontsize    is >&titlefontsize<;
        %put      titlerowheight   is >&titlerowheight<;
        %put      titleleadingspacecount is >&titleleadingspacecount<;
        %put      titletrailingspacecount is >&titletrailingspacecount<;
        %put      leadingrows      is >&leadingrows<;
        %put      leadingrowheight is >&leadingrowheight<;
        %put      columns          is >&columns<;
%do _cc = 1 %to &columns;
        %put      columnwidth&_cc  is >&&columnwidth&_cc<;
        %put      columnjustification&_cc is >&&columnjustification&_cc<;
%end;
        %put      footer           is >&footer<;
        %put      splitcharacter   is >&splitcharacter<;
        %put      outputmacro      is >&outputmacro<;
        %put      xlwinpath        is >&xlwinpath<;
        %put      columnnumbers    is >&columnnumbers<;
        %put      debug            is >&debug<;
        %put      RPTDEST macro arguments <end>: &syspbuff;
%mend;

```

```

*****
*                                     *
*   Macro code utilizing how the    *
*   macro can dynamically handle    *
*   varying number of arguments.   *
*                                     *
*****

```

```
%macro rptdest(Args) / pbuff;
```

```

%* use length() to avoid direct null string ;
%* test which can situationally incur error ;
%if %length(&syspbuff) gt 0 %then
%do;

        %* the incoming parameter buffer (syspbuff) ;
        %* contains the parenthesis around the    ;
        %* argument list -- strip those off      ;
        %let parmbuf = %substr(&syspbuff,2,%eval(%length(&syspbuff)-2));

        %* setup how far to loop ;
        %let parmlen = %length(&parmbuf);

        %* loop through the argument list ;
        %do _i = 1 %to &parmlen ;

                %* break the incoming parameter on commas ;
                %* (quoting the result so that special    ;

```

```

    %* characters do not interfere with the      ;
    %* scan function                          ;
%let _arg&i = %quote(%scan(%quote(&parmbuf),&i,','));

    %* now for some goofiness - the quoting    ;
    %* above will keep leading spaces, if the  ;
    %* macro was called with the message text  ;
    %* on multiple physical lines there are    ;
    %* a large number of leading spaces (up to ;
    %* nnn) that need to be trimmed off -- so, ;
    %* unquote the result (this will eliminate ;
    %* the leading spaces) and then quote it   ;
    %* again (to eliminate the meaning of      ;
    %* special characters) so that we can use  ;
    %* it in some macro functions below        ;
    %* (otherwise, things like paren- and      ;
    %* operation symbols screw up functions    ;
    %* when the variable is resolved)          ;
%let _arg&i = %unquote(&&_arg&i);
%let _arg&i = %quote(&&_arg&i);

    %* when we come to a empty value (the end) ;
    %* set the number of arguments found and    ;
    %* then set loop counter to eol to exit.    ;
%if &&_arg&i eq %then
%do;
    %let ArgCount = %eval(&i - 1);
    %let _i = &parmlen;
%end;
%else
%do;
    %* ignore all positional parameters for this application ;
    %if %sysfunc(index(%unquote(&&_arg&i),%str(=))) ne 0 %then
%do;
        %global %scan(&&_arg&i,1,'=');
        %* cannot just use scan2 for arg since may
        %* contain equals sign, adjust by 2 for quotes ;
        %if %length(&&_arg&i) ge %eval(%length(%scan(&&_arg&i,1,'='))+2) %then
%do;
            %let %scan(&&_arg&i,1,'=')=%substr(&&_arg&i,
                %eval(%length(%scan(&&_arg&i,1,'='))+2));
%end;
        %else
%do;
            %let %scan(&&_arg&i,1,'=');
%end;
%end;
%end;

%end;

%end;
%else
%do;
    %let Argcount=0;
%end;

    %* initialize ;
%_setdefault(data,&syslast);
%_setdefault(rundate,%sysfunc(today()),date9.);
%_setdefault(destination,prntrdy);
%_setdefault(xlfilename,rptdest);
%_setdefault(sheetname,Sheet1);
%_setdefault(orientation,portrait);
%_setdefault(fittopage,);

```

```

%_setdefault(fitwidth,1);
%_setdefault(fitheight,0);
%_setdefault(headermargin,0.5);
%_setdefault(topmargin,1.0);
%_setdefault(rightmargin,0.75);
%_setdefault(bottommargin,1.0);
%_setdefault(leftmargin,0.75);
%_setdefault(footermargin,0.5);
%_setdefault(titles,0);
%_setdefault(titlecolumn1,1);
%_setdefault(titlefontsize,12);
%_setdefault(titlerowheight,15.75);
%_setdefault(titleleadingspacecount,0);
%_setdefault(titletrailingspacecount,0);
%_setdefault(titlejustification1,c);
%_setdefault(leadingrows,0);
%_setdefault(leadingrowheight,12.75);
%_setdefault(columns,1);
%_setdefault(columnwidth1,8.43);
%_setdefault(columnjustification1,);
%_setdefault(footer,);
%_setdefault(splitcharacter,%str( ));
%_setdefault(outputmacro,);
%_setdefault(xlwinpath,);
%_setdefault(columnnumbers,);
%global _debug;
%_setdefault(debug,&_debug);

%* since multiple variables with indexes ;
%* and references can crossreference and ;
%* incrementally become cyclic, counters ;
%* are pretallied prior to the variables ;
%* being established to avoid references ;
%* to variable not being resolved. ;

%* it is very important that the maxidx, ;
%* maxref, autofill and makevars be kept ;
%* in the dependency order established. ;

%* establish max counter from indexes ;
%_maxidx(titles,title);
%_maxidx(titles,titlecolumn);
%_maxidx(titles,titlejustification);
%_maxidx(columns,columnwidth);
%_maxidx(columns,columnjustification);

%* establish max counter from references ;
%_maxref(columns,titlecolumn);

%* establish variables and autofill missings ;
%_autofill(titles,titlecolumn);
%_autofill(titles,titlejustification);
%_autofill(columns,columnwidth);

%* establish variables ;
%_makevars(titles,title);
%_makevars(columns,columnjustification);

%* compute default to title rows + leading rows + columnlabels;

```

```

_setdefault(rowstorepeat,%eval(&titles + &leadingrows + 1));

    /* apply leading/trailing spaces to titles if specified ;
    %_padtitles;

    /* derived variables and adjustments ;
    %if &fittopage ne %then %let fittopage=<x:FitToPage/>;
    %let rowstorepeat=&rowstorepeat; /* trim spaces ;
    %if &sysscp eq WIN %then
        %do;
            %if &xlwinpath ne %then
                %let xlwinpath=&xlwinpath\;
            %end;
        %else
            %do;
                %let xlwinpath=;
            %end;
        %end;

    %if &debug ne %then
        %do;
            %put NOTE: ArgCount is >&ArgCount<;
            /* display the arguments passed ;
            %do _i = 1 %to &ArgCount ;
                %put _arg&_i is >&&_arg&_i<;
            %end;
            %_shwvalues;
        %end;

    %if %upcase(&destination) eq CSV %then
        %do;
            %ds2csv(csvfref=_webout,
                data=&data,
                savefile=&xlwinpath&xlfilename..csv
            );
        %end;
    %else
        %do;

            %if &sysscp eq WIN %then
                %do;
                    filename htmlout "&xlwinpath&xlfilename..xls";
                    ods listing close;
                %end;
            %else
                %do;
                    %let rc=%sysfunc(appsrv_header(content-type,application/vnd.ms-
excel));
                    %let rc=%sysfunc(appsrv_header(content-
disposition,%str(attachment;filename=&xlfilename..xls)));
                %end;

            /* segments to keep footer line shorter ;
            %let ftL=Rundate\003A &rundate.;
            %let ftC=&footer.;
            %let ftR=Page %nrstr(&)P of %nrstr(&)N;

```

```

proc template;
  define tagset xlsx/ store=work;
    parent=tagsets.htmlcss;
    define event doc;
      start:
        put '<html xmlns:o="urn:schemas-microsoft-com:office:office"' NL;
        put '      xmlns:x="urn:schemas-microsoft-com:office:excel">' NL;
      finish:
        put "</html>" NL;
    end;

%if &sysssc eq WIN %then
  %do;
    define event system_title;
      put "<h1";
      trigger align;
      put ">";
      put VALUE / if !cmp(value,"The SAS System");
      put "</h1>" NL;
    end;

%end;

    define event doc_head;
      start:
        put "<head>" NL;
        put VALUE NL;
        put "<style>" NL;
        put "<!--" NL;
        trigger alignstyle;
        put '.style0' NL;
        put '  {mso-number-format:General;' NL;
        put '    text-align:general;' NL;
        put '    vertical-align:bottom;' NL;
        put '    white-space:nowrap;' NL;
        put '    mso-rotate:0;' NL;
        put '    mso-background-source:auto;' NL;
        put '    mso-pattern:auto;' NL;
        put '    color:windowtext;' NL;
        put '    font-size:10.0pt;' NL;
        put '    font-weight:400;' NL;
        put '    font-style:normal;' NL;
        put '    text-decoration:none;' NL;
        put '    font-family:Arial;' NL;
        put '    mso-generic-font-family:auto;' NL;
        put '    mso-font-charset:0;' NL;
        put '    border:none;' NL;
        put '    mso-protection:locked visible;' NL;
        put '    mso-style-name:Normal;' NL;
        put '    mso-style-id:0;}' NL;
        put '.ttlfnt' NL;
        put '  {mso-style-parent:style0;' NL;
        put '    font-size:&titlefontsize..0pt;' NL;
        put '    font-weight:700;' NL;
        put '    font-family:Arial, sans-serif;' NL;
        put '    mso-font-charset:0;' NL;
        put '    text-align:center;}' NL;
        put '.ttlfnt1' NL;
        put '  {mso-style-parent:style0;' NL;
        put '    font-size:&titlefontsize..0pt;' NL;
        put '    font-weight:700;' NL;
        put '    font-family:Arial, sans-serif;' NL;
        put '    mso-font-charset:0;' NL;
        put '    text-align:left;}' NL;
        put '.ttlfntc' NL;
        put '  {mso-style-parent:style0;' NL;
        put '    font-size:&titlefontsize..0pt;' NL;
        put '    font-weight:700;' NL;
        put '    font-family:Arial, sans-serif;' NL;
        put '    mso-font-charset:0;' NL;
        put '    text-align:center;}' NL;

```

```

        put ' .ttlfnt' NL;
        put ' {mso-style-parent:style0;' NL;
        put ' font-size:&titlefontsize..0pt;" NL;
        put ' font-weight:700;" NL;
        put ' font-family:Arial, sans-serif;" NL;
        put ' mso-font-charset:0;" NL;
        put ' text-align:right;}" NL;
        put "-->" NL;
        put "</style>" NL;
    finish:
        put "<style> @page{" NL;
        put " mso-footer-
data:""%nrstr(&)L%nrstr(&)9&ftL.%nrstr(&)C&ftC.%nrstr(&)R%nrstr(&)9&ftR."";" NL;

        put " mso-page-orientation: &orientation;" NL;
        put " margin: &topmargin.in &rightmargin.in
&bottommargin.in &leftmargin.in;" NL;
        put " mso-header-margin:&headermargin.in;" NL;
        put " mso-footer-margin:&footermargin.in;" NL;
        put ' mso-horizontal-page-align:center;}' NL;
        put '</style>' NL;
        put "<xml>" NL;
        put " <x:ExcelWorkbook>" NL;
        put " <x:ExcelWorksheets>" NL;
        put " <x:ExcelWorksheet>" NL;
        put " <x:Name>&sheetname.</x:Name>" NL;
        put " <x:WorksheetOptions>" NL;
        put " &fittopage." NL;
        put " <x:Print>" NL;
        put " <x:ValidPrinterInfo/>" NL;
        put " <x:FitWidth>&fitwidth.</x:FitWidth>" NL;
        put " <x:FitHeight>&fitheight.</x:FitHeight>" NL;
        put " </x:Print>" NL;
        put ' <x:FreezePanes/>' NL;
        put ' <x:FrozenNoSplit/>' NL;
        put "
<x:SplitHorizontal>&rowstorepeat.</x:SplitHorizontal>" NL;
        put "
<x:TopRowBottomPane>&rowstorepeat.</x:TopRowBottomPane>" NL;
        put " </x:WorksheetOptions>" NL;
        put " </x:ExcelWorksheet>" NL;
        put " </x:ExcelWorksheets>" NL;
        put " </x:ExcelWorkbook>" NL;
        put " <x:ExcelName>" NL;
        put " <x:Name>Print_Titles</x:Name>" NL;
        put " <x:SheetIndex>1</x:SheetIndex>" NL;
        put "
<x:Formula>=%str('%)&sheetname.%str('%)!$1:$&rowstorepeat.</x:Formula>" NL;
        put " </x:ExcelName>" NL;
        put "</xml>" NL;
        put "</head>" NL;
    end;
    define event table;
    start:
        trigger preimage;
        put "<table";
        putq " class=" HTMLCLASS;
        trigger align;
        trigger style_inline;
        putq " border=" BORDERWIDTH;
        put " border=""0"" / if !exist( BORDERWIDTH );
        putq " cellspacing=" CELLSPACING;
        putq " cellpadding=" CELLPADDING;
        putq " rules=groups";
        putq " frame=" FRAME;
        put NL;
        %do _cc = 1 %to &columns;
            %let columnwidth&_cc.=%sysfunc(round(%sysevalf(&&columnwidth&_cc. *
5.520995),1));
            %end;
            %do _tt = 1 %to &titles;

```

```

        put " <tr style=""height:&titlerowheight.pt"">" NL;
%do _cc = 1 %to &columns;
    %if &_cc eq &&titlecolumn&_tt.. %then
        %do;
            put " <td nowrap class=ttlfnt&&titlejustification&_tt..
style=""width:&&columnwidth&_cc..pt"" NL;
                put "
x:str=""'%quote(&&title&_tt..)'>%quote(&&title&_tt..)</td>" NL;
                    %end;
                %else
                    %do;
                        %if &_cc le &&titlecolumn&_tt.. %then
                            %do;
                                put " <td style=""width:&&columnwidth&_cc..pt""></td>" NL;
                                    %end;
                            %end;
                        %end;
                    %end;
                %end;
            put ' </tr>' NL;
        %end;
%if &leadingrows eq 0 and &columnnumbers ne %then
    %do;
        put ' <tr>' nl;
        %do _cc = 1 %to &columns;
            put " <th>&_cc.</th>" nl;
        %end;
        put ' </tr>' nl;
    %end;
%do _lr = 1 %to &leadingrows;
    %if &columnnumbers ne and &_lr eq 1 %then
        %do;
            put ' <tr>' nl;
            %do _cc = 1 %to &columns;
                put " <th>&_cc.</th>" nl;
            %end;
            put ' </tr>' nl;
        %end;
    %else
        %do;
            put " <tr style=""height:&leadingrowheight.pt""/>" NL;
        %end;
    %end;
%end;

finish:
    put '</table>' nl;
end;
define event data;
start:
    put '<th bgcolor=#DDDDFF ' / if cmp(section, "head");
    put '<td wrap valign=top ' / if !cmp(section, "head");
    trigger headalign / if cmp(section, "head");
%do _cc = 1 %to &columns;
    put " style=""width:&&columnwidth&_cc..pt "" " / if
cmp(COLSTART, "&_cc");
    %if &&columnjustification&_cc.. ne %then
        %do;
            %* column justification MUST be subsequent to trigger
headalign to override ;
            put " id=""&&columnjustification&_cc.."" " / if cmp(COLSTART,
"&_cc");
        %end;
    %end;
    trigger style_inline;
    trigger rowcol;
    put ">";
    put VALUE;
finish:
    put "</th>" nl / if cmp(section, "head");
    put "</td>" nl / if !cmp(section, "head");
end;
end;
run;

```

```

ods path sashelp.tmplmst work.work;

ods markup tagset=xlxml
%if &sysscp eq WIN %then
%do;
body=htmlout
%end;
%else
%do;
body=_webout
%end;
;

%if &outputmacro eq %then
%do;
proc print data=&data noobs width=uniform label split="&splitcharacter";
run;
%end;
%else
%do;
%&outputmacro;
%end;

ods markup close;

%end;

%mend;

*****
*
*                               End of Module
*
*****

```

CONCLUSIONS

This has been a huge labor-saving tool in application and report development for the particular style of reports that were specified. Your specifications may differ for the style and types of output but the point of this exercise is that once you create one working template is that you can find ways of turning it into something dynamic which may provide benefits in repeat use.

ACKNOWLEDGMENTS

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 registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Mark Terjeson

Email mterjeson@russell.com or mtnw@comcast.net