

Using Dynamic Data Exchange to Customize Formatted Reports in Microsoft® Excel

Dennis J. Beal, Science Applications International Corporation, Oak Ridge, TN

ABSTRACT

Dynamic data exchange (DDE) is a powerful tool within SAS® that programmers can use to send data stored in SAS data sets directly into formatted Microsoft Excel tables. This process both reduces human error because the data are not hand entered into Excel where numerical precision can be lost and saves time since the user need not format the table manually. This paper presents SAS code and macros for advanced SAS users that can be used to open Excel from SAS; center table titles across specified cells; bold, superscript, underline or italicize specific characters within individual cells; present numerical data in multiple formats such as scientific notation, commas or rounded to a specific precision; insert "page x of y" and date into the footer of the table; put borders around any column or row; dynamically assign rows and columns to insert data headings into the body of the table; load both numeric and character data into the same Excel column; specify font size and type; specify page setup options such as portrait, landscape and margins; autofit specific columns; and create multiple worksheets within a single Excel file in a specified order with sheet names specified by the user.

1. INTRODUCTION

The Output Delivery System (ODS) within SAS has many file formats into which SAS output can be exported. However, what if the SAS user wants a highly customized format for displaying data that ODS does not currently support? The answer is to use Dynamic Data Exchange (DDE). DDE is a communication protocol available on Windows and OS/2 that enables some applications on these platforms (e.g., Microsoft Access, Excel, Word, FoxPro and others) to communicate with each other in a client/server fashion. The client application initiates a conversation with the server application and asks the server to perform a specific task. SAS (version 6.08 and later) can act as a DDE-client but not as a server. Most things that can be done manually in Excel can be automated from within a base SAS program using DDE. DDE enables a SAS session to take control of the Excel application and tell it precisely what to do. More information about the general idea of using DDE between SAS and DDE-compliant applications is found in the Institute's Technical Support document TS325 (SAS Institute, 1999).

2. EXAMPLE DATA

In order to illustrate how flexible and powerful DDE can be used to create highly customized tables in Excel from within SAS, a fictitious data set was created. Due to space limitations the SAS program "assemble SESUG data.sas" used to read and summarize the environmental data and the complete raw data set cannot be displayed in this paper. The code and data set may be obtained from the author. Three media were sampled: sediment, surface soil and surface water. For each medium data were collected at two or three different locations for four quarters and analyzed for several types of chemical analysis: field measurements, inorganics, organic polychlorinated biphenyls (PCBs), organic volatiles and radionuclides. Environmental data are usually qualified by the laboratory that analyzes the environmental samples with qualifiers which determine whether the chemical was detected in the sample or not. A chemical that is not detected in the environmental sample is typically denoted with a "U" qualifier. If a "U" qualifier is not associated with the numeric result, then the result is said to be "detected." Detected results are then compared to current environmental criteria appropriate for each medium. Most chemicals have a maximum criterion that results should not exceed. However, the field measurement pH has an allowable range (6.5 to 9 standard units). Some chemicals do not have a criterion to compare against the data.

Tables 1 and 2 are the formatted summary tables in Excel for sediment and surface water that are created from the SAS code discussed in section 3. The frequency of detection for each chemical is shown in the tables as "x / y" where x denotes the number of detected results and y denotes the number (detected and not detected) of samples collected. The frequency that detected results meet or exceed the environmental criteria is shown as "x / y" where x denotes the number of detected results that meets or exceeds the criterion and y denotes the number of detected

results for that chemical. Some quarters have missing data for some chemicals. Therefore, the frequency of detection may only show 3 results instead of 4.

3. PREPARE DATA IN SAS

Before the data can be sent directly to Excel using DDE, the data set as it will appear in Excel should be created. In addition, several macros need to be defined that will be used in the commands SAS will send to Excel. Sections 3.1 through 3.9 describe these macros with SAS code. The SAS code presented in sections 3.1 – 3.9 uses the SAS System for personal computers version 8.2 (TS level 02M0) running on a Windows 2000 platform.

3.1 INITIAL MACROS

The SAS program begins by defining initial macros that tell SAS the name and location of the input data set, the subdirectory where the Excel file will be saved and name of the Excel file.

```
%let PATHDAT=C:\Projects\SAS programs\; ** subdirectory of input data set;
libname sesug "&PATHDAT"; ** create library name for directory;
%let PATHTAB=C:\Projects\SAS programs\; ** subdirectory for created Excel table;
%let INPDS = sesug.sesug_summary; ** input summarized data set name;
%let OUTPDS = outpds; ** SAS data set name for data sent to Excel;
%let FN = Summary tables; ** created Excel file with table (no ".xls");
```

The following macros store the number of Excel rows needed for the title (including the blank line after the title), the number of rows needed for the column headings, the maximum number of footnotes (including the blank line after the table and before the footnotes) needed for each table, the number of columns in the table and the estimated maximum number of pages each table will be.

```
%let TITLEHEAD = 2 ; ** rows needed for table title (including blank);
%let COLHEAD = 2 ; ** rows needed for column headings;
%let FOOTNOTE OBS = 10; ** rows needed for footnotes;
%let NUM_COLUMNS = 11; ** total number of columns for whole table;
%let MAX_PAGES = 15; ** estimated maximum number of pages per table;
```

The macro `obsnvars` is available in the SAS online documentation. A SAS data set is input and the macro assigns the number of variables and observations in the input data set to macro variables.

```
%macro obsnvars(ds);
  %global dset nvars nob;
  %let dset=&ds;
  %let dsid = %sysfunc(open(&dset));
  %if &dsid %then %do;
    %let nob = %sysfunc(attrn(&dsid,NOBS));
    %let nvars=%sysfunc(attrn(&dsid,NVARS));
    %let rc = %sysfunc(close(&dsid));
  %end;
  %else
    %put Open for data set &dset failed - %sysfunc(sysmsg());
  %mend obsnvars;
```

The macro `ds_names` is used in the `set` statement within the data step to set a specified number of data sets together with the same root name where the number of data sets is dynamically determined by the data. The two inputs are the data set name and the number of data sets to be included. The data set `title_&name` contains the heading row data (station designation or analysis type).

```
%macro ds_names(name, num);
  %local zz;
  %do zz = 1 %to &num;
    title_&name&zz. &name&zz.
  %end;
%mend ds_names;
```

For example, `%ds_names(a, 3)` will produce the statement `set title_a1 a1 title_a2 a2 title_a3 a3.`

3.2 DETERMINE THE NUMBER OF WORKSHEETS

The following SAS code determines the number of Excel worksheets that will be created (one table per worksheet for each of the three media). The SAS data set `media` contains the unique media names (sediment, surface soil and surface water). It is important that a `run;` statement appear before the `%obsnvars` macro call which stores the number of sheets in the macro variable `N_SHEETS`. The data set `matrix` creates the macro variables `MEDIA1`, `MEDIA2` and `MEDIA3` along with the matrix macro variables `MATRIX1`, `MATRIX2` and `MATRIX3`. The `MATRIXi` variables are used in the column headings to change when the media change worksheets.

```
proc sort data=&INPDS out=media(keep=media) nodupkey; by media; run;

%obsnvars(media); %let N_SHEETS = &nobs;

data matrix;
  length MATRIX $8;
  set media;
  if index(lowcase(media), 'water') > 0 then MATRIX = 'Water';
  else if index(lowcase(media), 'sediment') > 0 then MATRIX = 'Sediment';
  else if index(lowcase(media), 'soil') > 0 then MATRIX = 'Soil';
  call symput('MATRIX' || left(_N_), left(trim(matrix)));
  call symput('MEDIA' || left(_N_), left(trim(media))); run;
```

3.3 DEFINE MACRO FOR CREATING MULTIPLE WORKSHEETS

The macro `create_sheets` subsets the data by media and outputs each formatted data set to Excel. The `SHEETNUM` do loop goes backwards from `N_SHEETS` %to 1 %by -1 so the sheets are in numerical order in Excel. The macro variable `SHEETNAME` stores the media being processed and appears as the sheet name in Excel. The macro variable `PREV_SHEET` is defined at the end of the program as the current sheet name before the sheet name is changed in the `SHEETNUM` do loop. The data set `dat` subsets the data for the given media from the input data set. The data set `stations` contains all the unique stations for the given media. The number of unique stations is stored in the macro variable `N_STATIONS`. Each of the unique station names are stored in the macro variables `STATION1`, `STATION2`, etc. It is important that these station names are global macro variables.

```
%macro create_sheets;

%do SHEETNUM = &N_SHEETS %to 1 %by -1;
%global TABLE_NO SHEETNAME PREV_SHEET N_STATIONS N_DATASET;

%let TABLE_NO = &SHEETNUM.;
%let SHEETNAME = &&MEDIA&SHEETNUM.;    ** sheet name shown in tab in Excel **;

data dat;
  set &INPDS;
  where media="&SHEETNAME";

  proc sort data=dat out=stations(keep=station) nodupkey; by station; run;

  ** store number of unique stations for this media **;
  %obsnvars(stations); %let N_STATIONS = &nobs; run;

%do i = 1 %to &N_STATIONS;
  %global STATION&i.;
  %end;

data _null_;
  set stations;
  call symput('STATION' || left(_N_), left(trim(station))); run;
```

3.4 CREATE DATA SET FOR EXCEL

The macro `create_table` builds a SAS data as it will appear in Excel with header rows for both the stations and analysis types. The data set `station&N` further subsets the `dat` data set for each of the unique stations for this medium. The number of unique analysis types for this station is stored in the macro variable `N_STATION&N`. The unique analysis types are stored in the macro variables

STATION&N._ANATYPE1, STATION&N._ANATYPE2, etc. Data sets station&N._anatype&i. store the data for each station and analysis type. The data sets title_station&N._anatype&i. store the analysis type (inorganics, etc.) header row for the data in station&N._anatype&i. Additional variables bold, center and italic are used later as flags whether to bold, center or italicize the analysis type header row. The data set sta&N. uses the ds_names macro to set all the station&N._anatype&i. data sets together along with their analysis type header rows. Finally, the title_sta&N. data set stores the station header row for each station. The final data set with all header rows as it will appear in Excel is saved in the data set &OUTPDS. The variable row stores the row number as it will appear in Excel. The number of rows in the body of the Excel table is stored in the macro variable N_DATASET.

```

%macro create_table;

  %do N = 1 %to &N STATIONS;
    %global N_STATION&N._ANATYPES;

  data station&N.;
    set dat;
    where station = "&&STATION&N.";
    attrib _numeric_ format=best9. informat=best9.;

  proc sort data=station&N. out=anatypes(keep=media station anatype) nodupkey;
    by media station anatype; run;  ** run is very important here;

  %obsnvars(anatypes); %let N_STATION&N._ANATYPES = &nobs;

  %do i = 1 %to &&N_STATION&N._ANATYPES;
    %global STATION&N._ANATYPE&i.;
    %end;

  data _null_;
    set anatypes;
    call symput("STATION&N._ANATYPE"||left(_N_), left(trim(anatype))); run;

  %do i = 1 %to &&N_STATION&N._ANATYPES;
    data station&N._anatype&i.;
      set station&N.;
      where anatype="&&STATION&N._ANATYPE&i."; run;

    data title_station&N._anatype&i.;
      length CHEMICAL STATION $50 ANATYPE $ 30;
      STATION = "&&STATION&N.";
      ANATYPE = "&&STATION&N._ANATYPE&i.";
      CHEMICAL = "&&STATION&N._ANATYPE&i.";
      BOLD='N';  ** the anatype is not bolded;
      CENTER='N';  ** the anatype is not centered;
      ITALIC='Y';  ** the anatype is italicized; run;
    %end;

  ** "stack" all anatype data sets with their title rows within each station;
  data sta&N.;
    set %ds_names(station&N._anatype, &&N_STATION&N._ANATYPES); run;

  data title_sta&N.;
    length CHEMICAL STATION $50;
    STATION = "&&STATION&N.";
    CHEMICAL = "&&STATION&N.";
    BOLD='Y';  ** the station name is bolded;
    CENTER='Y';  ** the station name is centered;
    ITALIC='Y';  ** the station name is italicized; run;
  %end;
%mend create_table;

%create_table; run;

** "stack" all station data sets with their title rows for this media;
data &OUTPDS;

```

```

length CHEMICAL $50;
set %ds_names(sta, &N_STATIONS);
ROW = _N_ + &TITLEHEAD + &COLHEAD; ** stores the row data will be in Excel;
attrib _numeric_ format=best9. informat=best9.; run;

** store number of Excel rows in body of table;
%obsnvars(&OUTPDS); %let N_DATASET = &nobs;

```

3.5 DETERMINE ROWS FOR EXCEL TABLE

The macro `datarows` stores the first and last rows where data will appear in the table for each station and analysis type in the macro variables `STATION&i._MIN_ROW_ANATYPE&j.` and `STATION&i._MAX_ROW_ANATYPE&j.`. The Excel row where the analysis type head row will appear is stored in the macro variable `STATION&i._ANATYPE_HEADROW&j.`. The data sets `datarows&i._&j.` (for varying values of *i* and *j*) contain non-header row data that will be later loaded into the Excel table. The data sets `anagroup_rows&i._&j.` (for varying values of *i* and *j*) contain the analysis type header row data. The data set `headers_station` contains the station header row data. The rows the station headers will appear in the Excel table are stored in the macro variables `HEAD_STATION_ROW_GROUP&i.`

```

%macro datarows;
  %do i = 1 %to &N_STATIONS;
    %global HEAD_STATION_ROW_GROUP&i.;

    %do j = 1 %to &&N_STATION&i._ANATYPES;
      %global STATION&i._MIN_ROW_ANATYPE&j. STATION&i._MAX_ROW_ANATYPE&j.
        STATION&i._ANATYPE_HEADROW&j.;

data datarows&i._&j.;
  set &OUTPDS;
  where station="&&STATION&i." and anatype="&&STATION&i._ANATYPE&j." and bold=' ';

  proc sort data=datarows&i._&j.; by station anatype;
  proc summary data=datarows&i._&j.;
    var row;
    by station anatype;
    output out=rowmin&i._&j.(drop=_type_ _freq_) min=minrow max=maxrow; run;

data assign_dat_rows&i._&j.;
  set rowmin&i._&j.;
  call symput("STATION&i._MIN_ROW_ANATYPE&j.", minrow);
  call symput("STATION&i._MAX_ROW_ANATYPE&j.", maxrow); run;

data anagroup_rows&i._&j.;
  set &OUTPDS;
  where station="&&STATION&i." and chemical="&&STATION&i._ANATYPE&j." and bold='N';
  call symput("STATION&i._ANATYPE_HEADROW&j.", row); run;
%end;
%end;

data headers_station;
  set &OUTPDS;
  where bold='Y';
  call symput('HEAD_STATION_ROW_GROUP'||left(_N_), row);
  keep chemical bold row; run;
%mend datarows;

%datarows;

```

3.6 BOLD CELLS THAT EXCEED CRITERIA

The macro `bold` bolds all the contents of a single cell. The inputs to the macro are the row and column numbers for the cell and the font size. The macro `bold_all` calls the bold macro for each cell that needs bolding. The inputs to the macro `bold_all` are the data set name that contains the rows and columns of cells that will be bolded and the desired font size of the text. The SAS functions `%unquote` and `%str` allow the use of macro variables in the X4ML

commands. The data set `bolds` subsets the data for records that require bolding. Only the row and column variables are kept since this data set is used later in the `bold_all` macro call in section 4.12.

```
%macro bold(rnum, cnum, size);
data _null_;
  file sas2x1;
  put %unquote(%str(%'[select("r&rnum.c&cnum.")]'%)); ** select cell to bold;
  put %unquote(%str(%'[format.font("Arial",&size,true,false,false,1,
  false,false)]%')); ** third parameter is "true" for BOLD; run;
%mend bold;

%macro bold_all(dsname, size);
  %obsnvars(&dsname); %let TOTAL = &nobs;
  %if &TOTAL > 0 %then %do;
    data _null_;
      set &dsname;
      call symput('ROW' || left(_N_), row);
      call symput('COL' || left(_N_), col); run;
    %local i;
    %do i = 1 %to &total;
      %bold(&&ROW&i., &&COL&i., &size);
    %end;
  %end;
%mend bold_all;

data bolds;
  set &OUTPDS;
  if xcdsum > 0 then do;
    if (mind ^= . and mind <= criteria_min) or (mind >= criteria and criteria ^= .)
      then do; COL = 8; output; end; ** bold the minimum detected column;
    if (maxd ^= . and maxd <= criteria_min) or (maxd >= criteria and criteria ^= .)
      then do; COL = 9; output; end; ** bold the maximum detected column;
    COL = 10; output; ** bold the criteria column;
    COL = 11; output; ** bold the frequency > criteria column;
  end;
  keep row col; run; ** the variable ROW is defined in the data set &OUTPDS;

%global N_BOLDS;
%obsnvars(bolds); %let N_BOLDS = &nobs; run;
```

3.7 FORMAT IN SCIENTIFIC NOTATION

For very large or small numbers displayed in the table, scientific notation may be the preferred format. The data set `scientific` keeps the rows and columns of cells with numbers that are at least as large as $1.0E+6$ or smaller than $1.0E-3$. All numeric columns displayed in the table are checked. The macro `sci` formats a single cell in scientific notation. Its only inputs are the row and column numbers for the cell. The format can be changed to any number of decimal places such as "0.00E+00". The macro `sci_all` calls the `sci` macro for each cell that needs to be in scientific notation. The input to the macro `sci_all` is the data set `scientific`.

```
data scientific;
  set &OUTPDS;
  if ltmin ^=. and ltmin ^= 0 then do; if log10(abs(ltmin)) >= 6 or
    log10(abs(ltmin)) < -3 then do; COL = 4; output; end; end;
  if ltmax ^=. and ltmax ^= 0 then do; if log10(abs(ltmax)) >= 6 or
    log10(abs(ltmax)) < -3 then do; COL = 5; output; end; end;
  if mean ^=. and mean ^= 0 then do; if log10(abs(mean)) >= 6 or
    log10(abs(mean)) < -3 then do; COL = 6; output; end; end;
  if std ^=. and std ^= 0 then do; if log10(abs(std)) >= 6 or
    log10(abs(std)) < -3 then do; COL = 7; output; end; end;
  if mind ^=. and mind ^= 0 then do; if log10(abs(mind)) >= 6 or
    log10(abs(mind)) < -3 then do; COL = 8; output; end; end;
  if maxd ^=. and maxd ^= 0 then do; if log10(abs(maxd)) >= 6 or
    log10(abs(maxd)) < -3 then do; COL = 9; output; end; end;
  if criteria ^=. and criteria ^= 0 then do; if log10(abs(criteria)) >= 6 or
    log10(abs(criteria)) < -3 then do; COL = 10; output; end; end;
  keep row col; run;
```

```

%macro sci(rnum, cnum);
  data _null_;
    file sas2xl;
    put %unquote(%str(%'[select("r&rnum.c&cnum.")]%')); ** select cell;
    put '[format.number("0.0E+00")]'; run;
%mend sci;

%macro sci_all(dsname);
  %obsnvars(&dsname); %let TOTAL = &nobs;
  %if &TOTAL > 0 %then %do;
    data _null_;
      set &dsname;
      call symput('ROW' || left(_N_), row);
      call symput('COL' || left(_N_), col); run;
      %local i;
      %do i = 1 %to &total;
        %sci(&&ROW&i., &&COL&i.);
      %end;
    %end;
  %end;
%mend sci_all;

```

3.8 FORMAT WITH COMMAS

The data set `commas` contains rows and columns for cells with numeric data between 10,000 and 999,999 that will be displayed with commas. All numeric columns displayed in the table are checked. The macro `comma` formats a single cell with commas. The symbol '#' tells Excel to display a digit in that position if it exists, whereas "000,000" tells Excel to display leading zeros. The macro `comma_all` calls the `comma` macro for each cell requiring formats with commas. The input to the macro `comma_all` is the data set `commas`.

```

data commas;
  set &OUTPDS;
  if ltmin ^= 0 then do; if 4 <= log10(abs(ltmin)) < 6 then do; COL = 4; output;
  end; end;
  if ltmax ^= 0 then do; if 4 <= log10(abs(ltmax)) < 6 then do; COL = 5; output;
  end; end;
  if mean ^= 0 then do; if 4 <= log10(abs(mean)) < 6 then do; COL = 6; output; end;
  end;
  if std ^= 0 then do; if 4 <= log10(abs(std)) < 6 then do; COL = 7; output; end;
  end;
  if mind ^= 0 then do; if 4 <= log10(abs(mind)) < 6 then do; COL = 8; output; end;
  end;
  if maxd ^= 0 then do; if 4 <= log10(abs(maxd)) < 6 then do; COL = 9; output; end;
  end;
  if criteria ^= 0 then do; if 4 <= log10(abs(criteria)) < 6 then do; COL = 10;
  output; end; end;
  keep row col;

%macro comma(rnum, cnum);
  data _null_;
    file sas2xl;
    put %unquote(%str(%'[select("r&rnum.c&cnum.")]%'));
    put '[format.number("###,###")]'; run;
%mend comma;

%macro comma_all(dsname);
  %obsnvars(&dsname); %let TOTAL = &nobs;
  %if &TOTAL > 0 %then %do;
    data _null_;
      set &dsname;
      call symput('ROW' || left(_N_), row);
      call symput('COL' || left(_N_), col); run;
      %local i;
      %do i = 1 %to &TOTAL;
        %comma(&&ROW&i., &&COL&i.);
      %end;
    %end;
  %end;

```

```
%mend comma_all;
```

3.9 FORMAT TO 3 SIGNIFICANT DIGITS

Often one wants to see only a few significant digits in tables. The data set `dec3` keeps all rows and columns with numbers between 1 and 9.99. The macro `dec3` formats an individual cell to 3 significant digits. The format “0.0#” tells Excel to always display one digit to the left and right of the decimal point, but only display the last digit if one exists. For example, the number 9.30067 is displayed 9.3, whereas the number 1.0176 is displayed 1.02. However, the number with all its precision (defined in the `attrib varname format=best9. informat=best9.;` SAS statement) is seen in the formula bar in Excel when the cell is clicked on. The macro `dec3_all` calls the macro `dec3` for each cell requiring this format. The data set `dec3` is its only input. Other macros similar to `dec3` can also be defined to display other significant digits. For example, numbers between 10 and 99.9 can be formatted with “##.0” that displays the last digit for non-integers.

```
data dec3;
  set &OUTPDS;
  if ltmin ^= 0 then do; if 0 <= log10(abs(ltmin)) < 1 and int(ltmin) ^= ltmin and
    round(ltmin, 0.1) ^= ltmin then do; COL = 4; output; end; end;
  if ltmax ^= 0 then do; if 0 <= log10(abs(ltmax)) < 1 and int(ltmax) ^= ltmax and
    round(ltmax, 0.1) ^= ltmax then do; COL = 5; output; end; end;
  if mean ^= 0 then do; if 0 <= log10(abs(mean)) < 1 and int(mean) ^= mean and
    round(mean, 0.1) ^= mean then do; COL = 6; output; end; end;
  if std ^= 0 then do; if 0 <= log10(abs(std)) < 1 and int(std) ^= std and
    round(std, 0.1) ^= std then do; COL = 7; output; end; end;
  if mind ^= 0 then do; if 0 <= log10(abs(mind)) < 1 and int(mind) ^= mind and
    round(mind, 0.1) ^= mind then do; COL = 8; output; end; end;
  if maxd ^= 0 then do; if 0 <= log10(abs(maxd)) < 1 and int(maxd) ^= maxd and
    round(maxd, 0.1) ^= maxd then do; COL = 9; output; end; end;
  if criteria ^= 0 then do; if 0 <= log10(abs(criteria)) < 1 and int(criteria) ^=
    criteria and round(criteria, 0.1) ^= criteria then do; COL = 10; output; end;
  end;
  keep row col; run;

%macro dec3(rnum, cnum);
  data _null_;
  file sas2xl;
  put %unquote(%str(%'[select("r&rnum.c&cnum.")]'%));
  put '[format.number("0.0#")]'; run;
%mend dec3;

%macro dec3_all(dsname);
  %obsnvars(&dsname); %let TOTAL = &nobs;
  %if &TOTAL > 0 %then %do;
  data _null_;
  set &dsname;
  call symput('ROW' || left(_N_), row);
  call symput('COL' || left(_N_), col); run;
  %local i;
  %do i = 1 %to &total;
  %dec3(&&ROW&i., &&COL&i.);
  %end;
  %end;
%mend dec3_all;
```

4. SEND DATA TO EXCEL

Now that a data set has been created as it will appear in Excel and all the necessary formatting macros have been defined, we now focus on the specific commands used in SAS to send the formats and data to Excel. Our goal is to create from within SAS a high quality formatted table in Excel that is ready for publication or to be imported into Word. The server application is Microsoft Excel 2000 (9.0.6926 SP-3). Excel version 4 Macro Language (X4ML) is used to communicate the specific commands from SAS to Excel. Features that were added to Excel after version 4 may not have an X4ML equivalent. For example, SAS code in sections 4.1 – 4.14 will do all Excel formatting to create Tables 1 and 2 except for displaying the column headings at the top of every page. This feature was added after Excel version 4. This is the only task that must be completed manually in Excel after the table is created by selecting File → Page Setup → Sheet → Rows to repeat at top → \$1:\$4 where 4 is the number of rows to be repeated on every

page. All of the X4ML commands are located in the file "macrofun.exe" online at <http://www.microsoft.com/downloads/search.aspx?displaylang=en>. The executable file will create the "macrofun.hlp" help file that explains the full syntax of hundreds of X4ML functions, examples and cross-references to related commands and functions. A hard copy manual also exists for Excel 4 (Microsoft 1992). There is information that only appears in either the manual or in the help file, so having both should contain all the information needed for X4ML.

4.1 START EXCEL FROM SAS

The following SAS code launches Excel if it is not already running. If Excel is already running, the same SAS code will create a new file named in the macro variable FN defined in section 3.1. However, if an Excel file with the same name already exists in the macro variable PATHTAB subdirectory, this existing file will be overwritten. If an Excel file with the same name as FN is open from a different subdirectory than PATHTAB, a DDE error will occur because Excel cannot have two files with the same name open at the same time. The options statement uses the '@' symbol for missing values because when the X4ML function formula.replace("@", " ") is executed, it searches for the '@' symbol and replaces it with a blank. If the default missing symbol (a period ".") is used, Excel will replace the decimal point of each number with the specified symbol (blank in this case). The symbol used can be any symbol that will not be displayed in the Excel table. The filename sas2xl dde "excel|system" statement is a DDE system doublet that is used throughout the remaining SAS program and tells SAS the server application receiving the commands. The first _null_ data step launches Excel. The macro variable TAB is the hexadecimal tab that is used to move between columns in Excel. The macro variable S is used to delete the one trailing blank from character variables. The second _null_ data step creates a new Excel file with the name specified in the macro variable FN with one worksheet upon the first iteration of the SHEETNUM do loop. Subsequent iterations of the SHEETNUM loop insert a new worksheet before the previously created sheet and resave the file under the same name. In either case the error(false) command turns off the error checking within Excel and prevents Excel from displaying any message boxes that would stop the process until they are clicked away.

```
options noxsync noxwait xmin missing=@;
filename sas2xl dde 'excel|system';
data _null_;
  length fid rc start stop time 8;
  fid=fopen('sas2xl','s');
  if (fid le 0) then do;
    rc=system('start excel');
    start=datetime();
    stop=start+10;
    do while (fid le 0);
      fid=fopen('sas2xl','s');
      time=datetime();
      if (time ge stop) then fid=1;
    end;
  end;
  rc=fclose(fid); run;

%let tab='09'x;    * tab-character;
%let s+=(-1);     * use to delete trailing blank from character variables;

data _null_;
  file sas2xl;
  %if &SHEETNUM < &N SHEETS %then %do;
  ** opens existing file before inserting new sheet;
  put %unquote(%str(%[Open("&pathtab.&fn..xls",3)]%));
  put '[error(false)]'; ** switch off Excel error messages;
  ** activates the previously created sheet so new sheet will be inserted before it;
  put %unquote(%str(%[workbook.activate("&PREV_SHEET")]%));
  put '[workbook.insert()]'; ** insert another worksheet for next table;
  %end;
  %else %do;
  put '[new(1)]'; ** new workbook with 1 sheet;
  put '[error(false)]'; ** switch off Excel error messages;
  ** save the file with the given file name;
  put %unquote(%str(%[Save.as("&pathtab.&fn..xls")]%));
  %end; run;
```

4.2 MACROS FOR SELECTING ROWS AND CELLS

The `sel_rows` and `sel_cells` macros are simple but critically important macros that are used ubiquitously throughout the remaining SAS program. The `sel_rows` macro creates the DDE triplet filename reference `RECRANGE` for a specified range of rows and columns and tells Excel precisely where in the table to write data from a SAS data set. The `notab` option is a must because it will prevent the default conversion of blanks into delimiters within Excel.

```
%macro sel_rows(row_startno, col_startno, row_endno, col_endno);
    filename recrange dde "excel|[%fn.xls]sheet1!r&row_startno.c&col_startno.c
        &col_startno.:r&row_endno.c&col_endno." notab;
%mend sel_rows;
```

The `sel_cells` macro allows SAS macro variables to be used in selecting rows and columns for receiving formatting commands. For example, the macro call `%sel_cells(3, 1, 4, &NUM_COLUMNS);` selects rows 3 through 4 for columns 1 through `&NUM_COLUMNS` to receive the next X4ML formatting commands.

```
%macro sel_cells(row_startno, col_startno, row_endno, col_endno);
    put %unquote(%str(%'[select("r&row_startno.c&col_startno.:
        r&row_endno.c&col_endno.")]'%'));
%mend sel_cells;
```

The `fdate` macro formats the system date in any specified format. This date can be shown at the bottom of each table to show the table creation date. For example, the macro call `%fdate(mmddy10.)` puts the creation date in `mm/dd/yyyy` format.

```
%macro fdate(fmt);
    %global FDATE;
    data _null_;
        call symput("FDATE",left(put("&sysdate9"d,&fmt))); run;
%mend fdate;
%fdate(mmddy10.);
```

4.3 MACRO VARIABLES FOR ASSIGNING ROWS

Key rows from the table must be stored into macro variables that will be used as inputs to the `%sel_rows` macro. The macro variables are self-explanatory by their names. The `k` macro variables `PAGEBREAK_ROW&k.` store every row that will be the last row shown on each page of the table. The constant 69 is the number of rows that appear on a portrait table in Excel with Arial 9 point type font. This constant may need to be changed once the program is first run by viewing the first row with a dotted line underneath indicating a page break in Excel.

```
%macro rows;
    %local i k;
    %global FIRST_ROW_COL_HEAD TITLE_PLUS_COLHEAD FIRST_ROW_DATA
        LAST_ROW_DATA FIRST_ROW_FOOTNOTES LAST_ROW_FOOTNOTES;
    %let FIRST_ROW_COL_HEAD = %eval(&TITLEHEAD + 1);
    %let TITLE_PLUS_COLHEAD = %eval(&TITLEHEAD + &COLHEAD);
    %let FIRST_ROW_DATA = %eval(&TITLE_PLUS_COLHEAD + 1);
    %let LAST_ROW_DATA = %eval(&TITLE_PLUS_COLHEAD + &N_DATASET);
    %let FIRST_ROW_FOOTNOTES = %eval(&LAST_ROW_DATA + 1);
    %let LAST_ROW_FOOTNOTES = %eval(&LAST_ROW_DATA + &FOOTNOTE_OBS);
    %do i = 1 %to &FOOTNOTE_OBS;
        %global ROW &i._FOOTNOTES;
        %let ROW_&i._FOOTNOTES = %eval(&LAST_ROW_DATA + &i.);
    %end;
    %do k = 1 %to &MAX_PAGES;
        %global PAGEBREAK_ROW&k.;
        %let PAGEBREAK_ROW&k. = %eval(&TITLE_PLUS_COLHEAD+(69-&TITLE_PLUS_COLHEAD)*&k.);
    %end;
%mend rows;
%rows;
```

4.4 OVERRIDING EXCEL DEFAULT FORMATS

Excel has its own default formatting rules when it displays data. Often we want to use the formats we specify in SAS and not use the default formats in Excel. For example, the criterion for pH in our data set has a range from 6.5 to 9 standard pH units. If we send the text string "6.5 – 9" to Excel, Excel may interpret the dash between the numbers as a minus sign and subtract 9 from 6.5. Similarly, Excel will interpret the text string "3 / 4" for the frequency of

detection as the date 3/4/2004. The macro variable `N_PH_ROWS` contains the number of rows where the criteria must be presented as text.

```
data ph;
  set &OUTPDS;
  where criteria_txt ^= ' '; run;

  %obsnvars(ph); %let N_PH_ROWS = &nobs;

data _null_;
  set ph;
  call symput('PH_ROW'||left(_N_), row); run;
```

The symbol '@' in the following data step `_null_` tells Excel to leave the formatted SAS values alone and not to use the default Excel formats for the first row of data through the last row of footnotes for columns 1 through 3. Note no numeric data will appear in these first 3 columns. In addition, the last column showing the frequency of detected results exceeding the criteria also should not use the default Excel formats. Last, we tell Excel to use SAS formats for the rows where the criteria displayed should be text (such as pH) in column 10.

```
data _null_;
  file sas2xl;
  %sel_cells(&FIRST_ROW_DATA, 1, &LAST_ROW_FOOTNOTES, 3);
  put '[format.number("@")]'; ** rid Excel formats and use SAS formats;
  %sel_cells(&FIRST_ROW_DATA, &NUM_COLUMNS, &LAST_ROW_FOOTNOTES, &NUM_COLUMNS);
  put '[format.number("@)]';
  %do i = 1 %to &N_PH_ROWS;
    %sel_cells(&&PH_ROW&i., 10, &&PH_ROW&i., 10);
    put '[format.number("@)]';
  %end; run;
```

4.5 TABLE TITLE AND COLUMN HEADINGS

The title of the table is stored in the data step `lengths`. The length of the title is also calculated because a footnote will be shown at the end of the title. This adds flexibility to the program when the length of the title changes from one medium to the next. The length of the table title and the title are then stored in the macro variables `TITLE_LENGTH` and `TITLE`.

```
data lengths;
  ** the last 'a' is footnote;
  TITLE="Table &TABLE_NO.. Summary of Quarterly &SHEETNAME Dataa";
  ltitle=length(title); run;

data _null_;
  set lengths;
  call symput('TITLE_LENGTH', ltitle);
  call symput('TITLE', left(trim(title))); run;
```

The second `_null_` data step writes the title in the first row and column headings on rows 3 and 4. The `tab` macro variable is used to tab across columns when writing the column headings. The macro variable `MATRIX&SHEETNUM` allows the column heading for column 10 to change with the medium. There is an extra space at the end of each word in the last `put` statement so the headings on this row will look even on the table due to the superscripts in columns 6, 7, 10 and 11.

```
%sel_rows(1, 1, &TITLE_PLUS_COLHEAD, &NUM_COLUMNS);
data _null_;
  file rechange;
  put "&TITLE";
  put;
  put &tab &tab 'Frequency of' &tab 'Non-detects' &tab &tab &tab 'Std.' &tab
    'Detected' &tab &tab "&&MATRIX&SHEETNUM." &tab 'Frequency >';
  put 'Chemical ' &tab 'Units ' &tab 'Detection ' &tab 'Min ' &tab 'Max ' &tab
    'Meanb' &tab 'Dev.b' &tab 'Min ' &tab 'Max ' &tab 'Criteria' &tab 'Criteria';
  run;
```

4.6 STATION AND ANALYSIS TYPE HEADINGS

The macro `headings` places the formatted station and analysis type headings into the body of the table. The data sets and rows assigned to each heading were stored in macro variables in section 3.5. The X4ML function `border(,,1,1)` puts a single thin line over and under each station header row. The third parameter of the `format.font` function toggles the bold on while the fourth parameter toggles the italics on. The `alignment(7)` function centers the station heading across all columns, while `alignment(2)` left justifies the analysis type heading.

```
%macro headings;
  %local i j;
  %if &N_DATASET > 0 %then %do;
    %do i = 1 %to &N_STATIONS;
      %sel_rows(&&HEAD_STATION_ROW_GROUP&i., 1, &&HEAD_STATION_ROW_GROUP&i., 1);
      data _null_;
        set headers_station;
        if _N_=&i.;
        file rechange;
        put CHEMICAL; run;
      data _null_;
        file sas2xl;
        %sel_cells(&&HEAD_STATION_ROW_GROUP&i., 1, &&HEAD_STATION_ROW_GROUP&i.,
          &NUM_COLUMNS);
        put '[border(,,1,1)]';
        put '[format.font("Arial",9,true,true,false,false,1,false,false)]';
        put '[alignment(7)]'; ** 7=center across selection; run;
    %do j = 1 %to &&N_STATION&i._ANATYPES;
      %sel_rows(&&STATION&i._ANATYPE_HEADROW&j.,1,&&STATION&i._ANATYPE_HEADROW&j., 1);
      data _null_;
        set title_station&i._anatype&j.;
        file rechange;
        put CHEMICAL; run;
      data _null_;
        file sas2xl;
        %sel_cells(&&STATION&i._ANATYPE_HEADROW&j., 1, &&STATION&i._ANATYPE_HEADROW&j.,
          &NUM_COLUMNS);
        put '[format.font("Arial",9,false,true,false,false,1,false,false)]';
        put '[alignment(2)]'; ** 2=left justify across selection; run;
      %end;
    %end;
  %end;
%mend headings;
```

4.7 AUTOFIT COLUMNS

The macro `autofit` reduces the column widths for all columns to the minimum width so all data are visible. This macro is called towards the end of the program after all data have been written to the table.

```
%macro autofit(startno, endno);
  data _null_;
  file sas2xl;
  put %unquote(%str(%'[column.width(0,"c&startno.:c&endno.",false,3)]%')); run;
%mend autofit;
```

4.8 FORMAT DATA CELLS

The macro `borders_data` formats the data cells by replacing missing values with either blanks or dashes. The symbol in the `formula.replace` function need not be '@', but it must agree with the latest options `missing=` statement. For this project blanks are preferred for missing data in columns 4, 5, 8 and 9 while dashes are preferred for missing data in columns 6, 7 and 10. The data in the first column are left justified while data in the other columns are centered.

```
%macro borders_data(min_row, max_row);
  data _null_;
  file sas2xl;
```

```

%sel_cells(&MIN_ROW, 4, &MAX_ROW, 5);
put '[formula.replace("@", "-")]';
%sel_cells(&MIN_ROW, 6, &MAX_ROW, 7);
put '[formula.replace("@", "--")]';
%sel_cells(&MIN_ROW, 8, &MAX_ROW, 9);
put '[formula.replace("@", "-")]';
%sel_cells(&MIN_ROW, 10, &MAX_ROW, 10);
put '[formula.replace("@", "--")]';
%sel_cells(&MIN_ROW, 1, &MAX_ROW, 1);
put '[format.font("Arial",9,false,false,false,false,1,false,false)]';
put '[alignment(2)]'; ** 2=left justified;
%sel_cells(&MIN_ROW, 2, &MAX_ROW, &NUM_COLUMNS);
put '[format.font("Arial",9,false,false,false,false,1,false,false)]';
put '[alignment(7)]'; ** 7=center across selection; run;
%mend borders_data;

```

4.9 LOAD DATA

The macro `load_data` loads the data from the SAS data sets into the appropriate rows and columns defined in section 3.5 in the Excel table. Separate `put` statements are used depending on whether the criteria loaded is text or numeric. Two spaces are inserted before the chemical name to help offset the chemical names from the analysis type headers. The macro `TAB` is used to tab between columns. The `borders_data` macro is called to format the data after it is written into the table. If no data are in the data set, an alternate message of "No data to summarize." is printed.

```

%macro load_data;
  %local i j;
  %if &N_DATASET > 0 %then %do;
    %do i = 1 %to &N_STATIONS;
      %do j = 1 %to &&N_STATION&i._ANATYPES;
        %sel_rows(&&STATION&i._MIN_ROW_ANATYPE&j., 1,
                  &&STATION&i._MAX_ROW_ANATYPE&j., &NUM_COLUMNS);
        data load&i._&j.;
          set datarows&i._&j.;
          file rechange;
          if criteria_txt = ' ' then do;
            put ' ' CHEMICAL &s &tab UNITS &s &tab PDETC &s &tab LTMIN &tab LTMAX
              &tab MEAN &tab STD &tab MIND &tab MAXD &tab CRITERIA &tab PXCED;
          end;
          else do;
            put ' ' CHEMICAL &s &tab UNITS &s &tab PDETC &s &tab LTMIN &tab LTMAX
              &tab MEAN &tab STD &tab MIND &tab MAXD &tab CRITERIA_TXT &tab PXCED;
          end; run;
        %borders_data(&&STATION&i._MIN_ROW_ANATYPE&j., &&STATION&i._MAX_ROW_ANATYPE&j.);
      %end;
    %end;
  %end;

  %else %do;
    %sel_rows(&FIRST_ROW_DATA, 1, &FIRST_ROW_DATA, &NUM_COLUMNS);
    data _null_;
      file rechange;
      put "No data to summarize."; run;
    data _null_;
      file sas2xl;
      %sel_cells(&FIRST_ROW_DATA, 1, &FIRST_ROW_DATA, &NUM_COLUMNS);
      put '[format.font("Arial",10,false,false,false,false,1,false,false)]';
      put '[alignment(7)]'; ** 7=center; run;
    %end;
  %mend load_data;
%load_data;

```

4.10 FORMAT TITLE AND COLUMN HEADINGS

The following `_null_` data step performs several formatting functions for the title and column headings. The title is centered across all columns in Arial 10 point bold type. All column headings are also in Arial 10 point bold type.

Groups of individual cells are centered. For example, "r3c3" in the select function tells Excel to center the contents of the cell in row 3 column 3. "r3c4:r3c5" tells Excel to center the contents of the cells ("Non-detects") in columns 4 and 5 in row 3. Since headings exist in columns 10 and 11, "r3c10:r3c11" will center the contents of each cell individually.

```
data_null;
file sas2xl;

*** center title and specify fonts;
%sel_cells(1, 1, &TITLEHEAD, &NUM_COLUMNS);
put '[alignment(7)]'; ** 7=center across selection;
put '[format.font("Arial",9,true,false,false,false,1,false,false)]';

*** specify fonts and center column headings;
%sel_cells(&FIRST_ROW_COL_HEAD, 1, &TITLE_PLUS_COLHEAD, &NUM_COLUMNS);
put '[format.font("Arial",9,true,false,false,1,false,false)]';
put '[select("r3c3, r3c4:r3c5, r3c7, r3c8:r3c9, r3c10:r3c11")]';
put '[alignment(7)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 1, &TITLE_PLUS_COLHEAD, &NUM_COLUMNS);
put '[alignment(7)]';
```

The font.properties function superscripts 1 character beginning at position &TITLE_LENGTH of the title in row 1 column 1 using Arial Bold Italic type font. Similarly, the last character in each of the column headings on row 4 are superscripted.

```
put '[select("r1c1")]';
put %unquote(%str(%'[font.properties("Arial","Bold Italic",10,,true,,,,,,
&TITLE_LENGTH,1)]%'));
** superscript last character in column 1;
%sel_cells(&TITLE_PLUS_COLHEAD, 1, &TITLE_PLUS_COLHEAD, 1);
put '[font.properties("Arial",,9,true,,,,,,9,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 2, &TITLE_PLUS_COLHEAD, 2);
put '[font.properties("Arial",,9,true,,,,,,6,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 3, &TITLE_PLUS_COLHEAD, 3);
put '[font.properties("Arial",,9,true,,,,,,10,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 4, &TITLE_PLUS_COLHEAD, 5);
put '[font.properties("Arial",,9,true,,,,,,4,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 6, &TITLE_PLUS_COLHEAD, 7);
put '[font.properties("Arial","Bold Italic",9,true,,,,,,5,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 8, &TITLE_PLUS_COLHEAD, 9);
put '[font.properties("Arial",,9,true,,,,,,4,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 10, &TITLE_PLUS_COLHEAD, 11);
put '[font.properties("Arial","Bold Italic",9,true,,,,,,9,1)]';
```

The border function puts a single thin line (5th parameter = '1') under columns 4 and 5 in row 3 under "Non-detects" and under columns 8 and 9 in row 3 under "Detected." The border function puts a double line (5th parameter = '6') under all columns on row 4. The function puts single thin vertical lines (3rd parameter = '1') to the right of columns 3, 4, 5, 7, 8 and 9. The function puts a medium thick line (first parameter = '2') as an outline for the entire table.

```
** put borders around columns and rows;
%sel_cells(&FIRST_ROW_COL_HEAD, 4, &FIRST_ROW_COL_HEAD, 5);
put '[border(,,,1)]'; ** 1=single thin line under each row ;
%sel_cells(&FIRST_ROW_COL_HEAD, 8, &FIRST_ROW_COL_HEAD, 9);
put '[border(,,,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 1, &TITLE_PLUS_COLHEAD, &NUM_COLUMNS);
put '[border(,,,6)]'; ** 6=double line under row ;
%sel_cells(&FIRST_ROW_COL_HEAD, 3, &TITLE_PLUS_COLHEAD, 3);
put '[border(,1)]'; ** puts vertical lines after column 3;
%sel_cells(&FIRST_ROW_COL_HEAD, 5, &TITLE_PLUS_COLHEAD, 5);
put '[border(,1)]';
%sel_cells(&FIRST_ROW_COL_HEAD, 7, &TITLE_PLUS_COLHEAD, 7);
put '[border(,1)]';
%sel_cells(&FIRST_ROW_COL_HEAD, 9, &TITLE_PLUS_COLHEAD, 9);
put '[border(,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 4, &TITLE_PLUS_COLHEAD, 4);
put '[border(,1)]';
```

```

%sel_cells(&TITLE_PLUS_COLHEAD, 8, &TITLE_PLUS_COLHEAD, 8);
put '[border(,,1)]';
%sel_cells(&FIRST_ROW_COL_HEAD, 1, &LAST_ROW_DATA, &NUM_COLUMNS);
put '[border(2)]'; ** puts medium thick line as outline for entire table;

```

4.11 SET MARGINS, HEADERS AND FOOTERS

The `page.setup` function specifies the left, right, top and bottom margins, portrait or landscape orientation, footers and headers. The font size, left or right justification, font type and text for the header are specified in the first parameter, while the second parameter controls these same specifications for the footer. The eleventh parameter is a '2' for landscape orientation and '1' for portrait. The thirteenth parameter is the scale number representing the percentage to increase or decrease the size of the sheet. The creation date is hard coded using the `fdate` macro defined in section 4.2.

```

*** put in headers and footers and set margins ;
put %unquote(%str(%' [page.setup(%nrquote("&9&R&"Arial""DRAFT"),%nrquote("&9&C&"
"Arial""Page &P of &N&9&R&"Arial")"&FDATE",1,1,1,1,,false,true,false,2,,100)]%'));
run;

```

4.12 LAST FORMATS

The `last_formats` macro invokes several macros defined in sections 3.6, 3.7, 3.8, 3.9, 4.6 and 4.7. It is important to autofit the first column before inserting the analysis type group headings since they may be longer than the chemical names. Then the `autofit` macro can be invoked for the remaining columns after the numeric cells in these columns have been formatted. The second half of the `last_formats` macro draws a medium thick line under the last row of data appearing on each page, in addition to placing a medium thick line above the column headings.

```

%macro last_formats;

  %if &N_DATASET > 0 %then %do;
    %autofit(1, 1);
    %headings;
    %bold_all(bolds, 9);
    %sci_all(scientific);
    %dec0_all(dec0);
    %dec1_all(dec1);
    %dec2_all(dec2);
    %dec3_all(dec3);
    %dec4_all(dec4);
    %dec5_all(dec5);
    %dec6_all(dec6);
    %comma_all(commas);
    %autofit(2, &NUM_COLUMNS);
  %end;

  data _null_;
  file sas2xl;
  %do k = 1 %to &MAX_PAGES;
    %if &N_DATASET >= &&PAGEBREAK_ROW&k. - &TITLE_PLUS_COLHEAD %then %do;
      %sel_cells(&&PAGEBREAK_ROW&k., 1, &&PAGEBREAK_ROW&k., &NUM_COLUMNS);
      put '[border(,,,2)]'; ** puts medium thick line under last row on each page;
    %end;
  %end;

  %sel_cells(&TITLEHEAD, 1, &TITLEHEAD, &NUM_COLUMNS);
  put '[border(,,,2)]'; ** puts medium thick line on top of column headings;
  %sel_cells(&LAST_ROW_DATA, 1, &LAST_ROW_DATA, &NUM_COLUMNS);
  put '[border(,,,2)]'; ** puts medium thick line under last row of data; run;
%mend last_formats;
%last_formats;

```

4.13 FOOTNOTES

The `put_footnotes` macro outputs footnotes at the end of the table. The footnotes can be formatted with the footnotes in superscript italic font while specific words can be italicized, bolded or underlined within the same cell. Some footnotes can be output only if a criterion is met. For example, the footnote stating that results exceeding

criteria are shown in bold is output only if there are cells bolded in that table by using the `N_BOLDS` macro variable. The first `_null_data` step outputs the text of the footnotes while the second `_null_data` step formats specific words or letters of the footnotes.

```
%macro put_footnotes;
%sel_rows(&FIRST_ROW_FOOTNOTES, 1, &LAST_ROW_FOOTNOTES, &NUM_COLUMNS);
data _null_;
  file recrange;
  put;
  put 'a Fictional data are shown here for the sole purpose of this paper.';
  put 'b Mean and standard deviation are shown to 3 significant digits or the
    nearest integer. Nondetected results were used as reported.';
  put 'c Criteria taken from Table 4.3 in County Handbook of Environmental Criteria
    by Howard Sprague (http://www.countycriteria.gov).';
  put %unquote(%str('%-- Criteria not available or insufficient data to calculate
    the summary statistic.%'));
%if &N_BOLDS > 0 %then %do;
  put 'Results that exceed criteria are shown in bold.';
%end;
  put 'PCBs = polychlorinated biphenyls.'; run;

data _null_;      ** set fonts for footnotes;
  file sas2xl;
  %sel_cells(&FIRST_ROW_FOOTNOTES, 1, &LAST_ROW_FOOTNOTES, &NUM_COLUMNS);
  put '[format.font("Arial",8,false,false,false,1,false,false)]';
  put '[alignment(2)]'; ** 2=left justified;
  %sel_cells(&ROW_2_FOOTNOTES, 1, &ROW_4_FOOTNOTES, 1);
  ** superscript footnote in italic Arial font at character position 1;
  put '[font.properties("Arial","Italic",8,,true,,,,,1,1)]';
  %sel_cells(&ROW_4_FOOTNOTES, 1, &ROW_4_FOOTNOTES, 1);
  ** italicize name of document beginning in position 36 for 41 chars long;
  put '[font.properties("Arial","Italic",8,,,,,36,41)]';
  ** underline URL beginning in position 97 for 29 characters long;
  put '[font.properties("Arial",,8,,,,,true,,,97,29)]';
%if &N_BOLDS > 0 %then %do;
  %sel_cells(&ROW_6_FOOTNOTES, 1, &ROW_6_FOOTNOTES, 1);
  put '[font.properties("Arial","Bold",8,,,,,43,4)]'; ** bold the word BOLD;
%end; run;
%mend put_footnotes;
%put_footnotes;
```

4.14 RENAME THE WORKSHEET

After all formatting has been completed, it is time to rename the worksheet. Until now the worksheet has been assigned the default name "sheet1". The sheet is renamed to the macro variable `SHEETNAME` defined in section 3.3. An Excel macro must be used to rename the worksheet. Before the `%do` loop is incremented, the current sheet name is stored in the macro variable `PREV_SHEET` before the new sheet name is assigned for the next table.

```
data _null_;
  file sas2xl;
  put '[workbook.next()]'; ** be sure next sheet (only 1) is selected in Excel;
  put '[workbook.insert(3)]'; run; ** 3=insert Excel macro variable;

filename xlmacro dde 'excel|macro1!r1c1:r1000c1' notab lrecl=200;
data _null_;
  file xlmacro;
  put %unquote(%str('%=workbook.name("sheet1",&SHEETNAME.)%')); ** rename sheet;
  put '=halt(true)';
  put '!dde flush';
  file sas2xl;
  put '[run("macro1!r1c1")]'; run;

data _null_;
  file sas2xl;
  put '[workbook.delete("macro1")]';
  put '[save()]';
```

```

put %unquote(%str(%'[workbook.activate("&SHEETNAME.")]'%));
put '[select("r1c1")]'; run;

```

```

%let PREV_SHEET = &SHEETNAME; ** store current sheet name as previous sheet;
%end; ** end SHEETNUM do loop;
%mend create_sheets;
%create_sheets;

```

5. TABLE FORMATTED

Tables 1 and 2 are the formatted tables in Excel created directly from SAS.

Table 1. Summary of Quarterly Sediment Data^a

Chemical	Units	Frequency of Detection	Non-detects		Mean ^b	Std. Dev. ^b	Detected		Sediment Criteria ^c	Frequency > Criteria ^c
			Min	Max			Min	Max		
Great Dismal Swamp										
<i>Field Measurements</i>										
pH	Std Unit	4 / 4			7.6	0.627	6.9	8.4	6.5 - 9	0 / 4
<i>Inorganics</i>										
Aluminum	mg/kg	0 / 4	2010	2908	2447	383			5000	0 / 0
Lead	mg/kg	4 / 4			511	249	205	747	50	4 / 4
Mercury	mg/kg	2 / 4	10	17.5	279	430	175	913	25	2 / 2
<i>Organic PCBs</i>										
PCB-1254	mg/kg	1 / 4	6	10	20.8	25.6	59	59	20	1 / 1
PCB-1260	mg/kg	2 / 4	10	10	36	45.0	21	103	20	2 / 2
<i>Organic Volatiles</i>										
Benzene	mg/kg	1 / 4	19	25	75.3	107	236	236	25	1 / 1
Xylenes	mg/kg	0 / 4	1.0E-06	5.0E-06	3.3E-06	1.7E-06			1.0E-05	0 / 0
<i>Radionuclides</i>										
Gross Alpha	pCi/g	4 / 4			300	373	75	855	--	--
Gross Beta	pCi/g	2 / 4	0.5	0.9	76.1	118	54	249	50	2 / 2
R.E. Lee Natural Bridge										
<i>Field Measurements</i>										
pH	Std Unit	4 / 4			7.4	1.46	5.9	9.4	6.5 - 9	2 / 4
<i>Inorganics</i>										
Aluminum	mg/kg	1 / 4	2010	2570	8990	13,395	29,080	29,080	5000	1 / 1
Lead	mg/kg	2 / 4	20	21	207	360	41	747	50	1 / 2
Mercury	mg/kg	1 / 4	9.3	17.5	53.0	81.5	175	175	25	1 / 1
<i>Organic PCBs</i>										
PCB-1254	mg/kg	1 / 4	4	10	20.5	25.8	59	59	20	1 / 1
PCB-1260	mg/kg	2 / 4	10	10	42	52.1	29	119	20	2 / 2
<i>Organic Volatiles</i>										
Benzene	mg/kg	2 / 4	19	20	204.5	253	223	556	25	2 / 2
Xylenes	mg/kg	1 / 4	4.0E-06	1.0E-04	0.0075	0.015	0.03	0.03	1.0E-05	1 / 1
<i>Radionuclides</i>										
Gross Alpha	pCi/g	3 / 4	10	10	265	401	15	855	--	--
Gross Beta	pCi/g	3 / 4	0.5	0.5	113	161	19	349	50	2 / 3

^a Fictional data shown here for the sole purpose of this paper.

^b Mean and standard deviation are shown to 3 significant digits or the nearest integer. Nondetected results were used as reported.

^c Criteria taken from Table 4.3 in *County Handbook of Environmental Criteria* by Howard Sprague (<http://www.countycriteria.gov>).

-- Criteria not available or insufficient data to calculate the summary statistic.

Results that exceed criteria are shown in **bold**.

PCBs = polychlorinated biphenyls.

Table 2. Summary of Quarterly Surface Water Data^a

Chemical	Units	Frequency of Detection	Non-detects		Mean ^b	Std. Dev. ^b	Detected		Water Criteria ^c	Frequency > Criteria ^c
			Min	Max			Min	Max		
Myers Lake										
<i>Field Measurements</i>										
Conductivity	umho/cm	4 / 4			269.5	162	159	506	500	1 / 4
Temperature	deg C	4 / 4			19.3	5.31	12.3	25	--	--
pH	Std Units	4 / 4			6.45	1.52	4.2	7.5	6.5 - 9	1 / 4
<i>Inorganics</i>										
Copper	mg/L	4 / 4			77.5	20.1	51	95	50	4 / 4
Mercury	mg/L	0 / 4	9.9	19	14.0	4.72			25	0 / 0
<i>Organic PCBs</i>										
PCB-1254	mg/L	1 / 4	5	15	30.3	40.7	91	91	20	1 / 1
PCB-1260	mg/L	1 / 4	10	19	17.5	9.26	30	30	20	1 / 1
<i>Organic Volatiles</i>										
Benzene	mg/L	1 / 3	20	22	32.3	19.7	55	55	25	1 / 1
Xylenes	mg/L	1 / 4	6.0E-07	2.0E-04	0.35	0.7	1.4	1.4	1.0E-05	1 / 1
<i>Radionuclides</i>										
Gross Alpha	pCi/L	3 / 4	0.5	0.5	21.1	24.8	11	57	--	--
Gross Beta	pCi/L	2 / 4	0.7	0.9	27.7	45.9	13	96	50	1 / 2
Old Man Kelseys Creek										
<i>Inorganics</i>										
Mercury	mg/L	1 / 4	6.9	10	250,007	499,996	1.0E+6	1.0E+6	25	1 / 1
Nickel	mg/L	3 / 4	10	10	261	204	216	497	--	--
<i>Organic PCBs</i>										
PCB-1254	mg/L	1 / 4	7	15	22.8	24.4	59	59	20	1 / 1
PCB-1260	mg/L	0 / 4	10	17	14	3.16			20	0 / 0
<i>Organic Volatiles</i>										
Benzene	mg/L	0 / 4	20	26	22.8	3.2			25	0 / 0
Xylenes	mg/L	0 / 4	7.0E-07	8.0E-05	2.1E-05	3.9E-05			1.0E-05	0 / 0
Methylbenzene	mg/L	3 / 4	110	110	1309	1363	240	2901	1000	2 / 3
<i>Radionuclides</i>										
Gross Alpha	pCi/L	1 / 4	0.3	1.8	25.4	48.4	98	98	--	--
Gross Beta	pCi/L	3 / 4	0.4	0.4	2413	4812	9.1	9631	50	1 / 3
Tuckers Lake										
<i>Field Measurements</i>										
Conductivity	umho/cm	4 / 4			281	144	134	445	500	0 / 4
Temperature	deg C	4 / 4			18.8	5.89	11.3	24	--	--
pH	Std Units	4 / 4			7.1	0.294	6.8	7.5	6.5 - 9	0 / 4
<i>Inorganics</i>										
Lead	mg/L	4 / 4			25,021	49,986	10	100,000	50	1 / 4
Mercury	mg/L	1 / 4	8.9	13	22.7	24.2	59	59	25	1 / 1
<i>Organic Volatiles</i>										
Benzene	mg/L	1 / 4	20	26	37.8	31.6	85	85	25	1 / 1
Xylenes	mg/L	0 / 4	7.0E-07	2.0E-05	7.4E-06	9.1E-06			1.0E-05	0 / 0
<i>Radionuclides</i>										
Gross Alpha	pCi/L	2 / 4	0.4	1	10.6	12.3	15	26	--	--
Gross Beta	pCi/L	3 / 4	0.4	0.4	248	477	9.9	963	50	1 / 3

^a Fictional data shown here for the sole purpose of this paper.

^b Mean and standard deviation are shown to 3 significant digits or the nearest integer. Nondetected results were used as reported.

^c Criteria taken from Table 4.3 in *County Handbook of Environmental Criteria* by Howard Sprague (<http://www.countycriteria.gov>).

-- Criteria not available or insufficient data to calculate the summary statistic.

Results that exceed criteria are shown in **bold**.

PCBs = polychlorinated biphenyls.

6. CONCLUSION

This paper has shown an example SAS program that uses Dynamic Data Exchange (DDE) to create highly customized, formatted tables in Excel. Although more SAS code is usually needed to produce a customized table compared to ODS, the advantages over ODS are many. First, the potential for human error while formatting an Excel table manually is virtually eliminated since results are output directly from SAS into a high quality, report-ready table. Second, the time efficiency gained by having SAS produce multiple tables on multiple worksheets within the same Excel file is impressive. Third, DDE gives the SAS user complete control over how the table will be formatted. Finally, with the use of SAS macro variables there is practically no limit to the types of formatted tables that can be created in Excel from SAS. Once the SAS program is operational it can easily be adapted to produce similar tables for other projects and applications to gain even more efficiency.

REFERENCES

- Bodt, M. (1996), "Talking to PC Applications Using Dynamic Data Exchange," *Observations*, 5(3), 18-27.
- Microsoft Corporation (1992), *Microsoft Excel Function Reference*, Document Number AB26298-0592.
- SAS Institute (1999), "Technical Support Document #325 – The SAS System and DDE" <<http://ftp.sas.com/techsup/download/technote/ts325.pdf>> (July 19, 2004).
- Schreier, H. (1998), "Getting Started with Dynamic Data Exchange," *Proceedings of the Sixth Annual Southeastern SAS Users Group Conference*, 207-215.
- Vyverman, Koen (2000), "Using Dynamic Data Exchange to Pour SAS Data into Microsoft Excel," *Proceedings of the 18th SAS European Users Group International Conference*.
- Vyverman, Koen (2002), "Using Dynamic Data Exchange to Export Your SAS Data to MS Excel – Against All ODS, Part I," *Proceedings of the 27th Annual SAS Users Group Conference*.

CONTACT INFORMATION

The author welcomes and encourages any questions, corrections, feedback and remarks. Contact the author at:

Dennis J. Beal
Statistician / Risk Scientist
Science Applications International Corporation
P.O. Box 2501
151 Lafayette Drive
Oak Ridge, Tennessee 37831
phone: 865-481-8736
fax: 865-481-8714
e-mail: dennis.j.beal@saic.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 registered trademarks or trademarks of their respective companies.