Paper 238-2012

Copy and Paste Almost Anything

Arthur S. Tabachneck, Myqna, Inc., Thornhill, Ontario (Canada)
Randy Herbison, Westat, Rockville, MD
John King, Ouachita Clinical Data Services, Inc., Mount Ida, AR
Richard A. DeVenezia, Independent Consultant, Remsen, NY
Nate Derby, Stakana Analytics, Seattle, WA
Ben Powell, Genworth Financial, London, England

ABSTRACT

Every day, data appear on computer screens in the form of spreadsheets, wiki pages, HTML, PDF, Word documents or any of the methods that are used to display data forms and tables. And, with any of those formats you can typically highlight and copy only the data you desire to your computer's clipboard. However, because there currently isn't a PROC IMPORT dbms=clipbrd option, how can you paste such data into a SAS® dataset? The present paper provides code that we believe can be used to accomplish most such tasks and, at the same time, provides examples of features that we think should be available in PROC IMPORT for all DBMS options.

BACKGROUND

The present paper began as an effort to see if datastep techniques could be used to create a non *ad hoc* PROC IMPORT-like program that one might be able to use for importing data from their system's clipboard. Like many of the current authors' papers, the question was raised on either SAS-L or the SAS Discussion Forums. And, like many of the questions raised on such forums, the question can either be answered with code designed for a specific purpose or, as in the present case, more generalizable code can be offered.

SAS has provided the clipbrd access method since at least Version 6, but it appears to only be accessible via either datastep code or SAS/AF, does not include most of the options provided with other methods, and has an inherent limitation in that tabs are converted into a series of spaces. However, the method still appears to be capable of allowing one to create datastep code that accomplishes many of the same tasks that PROC IMPORT provides for other access methods.

PROC IMPORT accomplishes a number of background tasks that many of us might take for granted. For example, variable names are either automatically assigned or extracted from the input data, are modified to ensure that they represent valid SAS variable names and, additionally, are stored as variable labels. Data formats and informats, similarly, are selected and assigned based on the patterns that are found to exist in the input data. And, as needed to import a particular type of data that PROC IMPORT was designed to facilitate, various options are available for users to indicate their specific requirements.

PURPOSE

The purpose of the present effort was to create code that, without modification, could be used to import data from a system's clipboard. Like PROC IMPORT, which provides users with different options based on the types of files they are trying to import, the code presented in this paper was designed to let users paste four different types of data tables and specify their desired options for each. Two of those types appear to be identical on one's monitor, namely with variable names across the top row of the table, followed by any number of rows which contain values for each variable.

An example of such a table is shown in Figure 1 on the following page. If one were to create a SAS dataset based on the last two rows of that table, they might create a file labeled "tenure ", with eight variables, namely one for "Type" and one for each of the seven categories for which percentages are displayed.

However, before one can import such a table, they must copy it using a browser or software that maintains the horizontal tabs which separate the table's columns. Google Chrome appears to do this quite reliably for non-pdf files, Internet Explorer requires one to click on file->edit with Microsoft Word before being able to do it successfully, and version 6 of Adobe Acrobat Reader with a free downloadable add-on appears to work quite well for pdf files.

Data	All households	Lowest fifth	Second fifth	Middle fifth	Fourth fifth	Highest fifth	Top 5%
Households (in 1000s)	113,146	22,629	22,629	22,629	22,629	22,629	5,695
Lower limit	$0	$0	$18,500	$34,738	$55,331	$88,030	$157,176
Median number of income earners	1	0	1	1	2	2	2
Tenure							
Owner occupied	62.4%	49.0%	58.8%	68.9%	80.5%	90.0%	92.8%
Renter occupied	29.2%	48.3%	39.7%	29.9%	18.7%	9.6%	6.9%

Source: http://en.wikipedia.org/wiki/Household_income_in_the_United_States

Figure 1
A type of table commonly found on the web

For example, when the file shown in Figure 1 is highlighted and copied using one of those methods, and then pasted into a text editor (like notepad), it might appear in a text editor as shown in Figure 2.

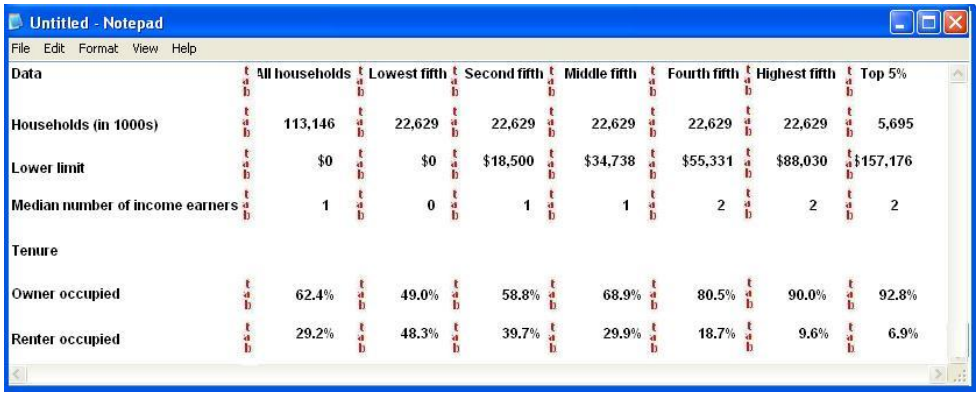
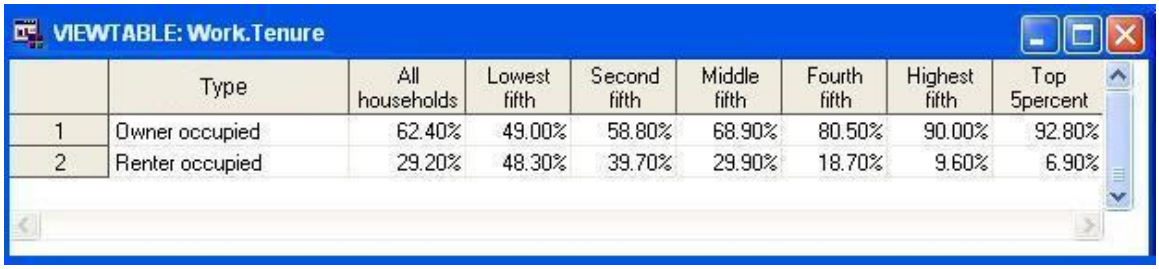


Figure 2
The way such a file really looks after being copied to a system's clipboard

Specifically, as shown in Figure 2, the clipboard will contain a tab-delimited file with variable names shown in the first record, and the data shown in the records that follow the variable names.

Now, if you were asked to create the SAS dataset shown in Figure 3, where would you begin (other than asking some very nice, careful person to re-enter all of the data)?



	Type	All households	Lowest fifth	Second fifth	Middle fifth	Fourth fifth	Highest fifth	Top 5percent
1	Owner occupied	62.40%	49.00%	58.80%	68.90%	80.50%	90.00%	92.80%
2	Renter occupied	29.20%	48.30%	39.70%	29.90%	18.70%	9.60%	6.90%

Figure 3
The SAS file you really want

In this particular situation, if one were to paste the table in a text editor (like Notepad), and save it with a .txt extension, PROC IMPORT could have correctly done all of the desired tasks except for renaming the first variable name. Conversely, if one were to paste the table into, and save the file as, an Excel spreadsheet, PROC IMPORT would necessarily import all seven rows and the variable formatting would be lost.

With the code provided in the present paper the file you can be imported directly from your system's clipboard, only import the desired rows, and be able to change the first variable's name without having to write and run any additional datasteps.

A table that looks quite similar, but which PROC IMPORT can't correctly import even with the help of a text editor, is one like that found at http://www.thelawyer.com/directory/uk-200-table-top-100/ and shown, below, in Figure 4. If you were to use your mouse to highlight the table, you would notice that all 101 rows get highlighted for each column separately. If you were to then paste the contents of the clipboard into a text editor (like Notepad), you would discover a single column, 404 row table. However, this type of table can also be imported directly using the code provided in the present paper. Additionally, if you wanted to change the two right most variable names to "Revenue" and "PEP", and convert the data to represent the actual units of measurement instead of simply indicating those units in the variable names, both tasks could easily be accomplished with the code provided in the current paper.

Figure 4
A table that copies as a long single column

A third type of table one might encounter is one that needs to be transposed in order to result in a meaningful SAS dataset. An example of such a table would be one where the variable names are provided in the first column and the data for each variable provided in the remaining columns. To import such a table using PROC IMPORT, one would have to save the file in one or another format, import it, transpose the file using the information in the first column as an ID, and then include a datastep to create new variables that reflect the actual data (e.g., numeric rather than character) and apply the desired formats. Again, using the code provided in the present paper, accomplishing such a task can be done quite easily.

A fourth type of data one often encounters is the form, or structured printout, as shown in Figure 5 on the following page. Such files not only pose all of the complexities of the other three types, but can also contain such complexities as multiple fields per row, multiple and possibly even variable numbers of rows per field, and boundary locations and/or keystrings. The example shown in Figure 5 is a rather simple example of such a form, namely the result of a library search for books that have been written by Stephen King.

If one were to copy and paste the table in a text editor, they would likely see a file that has seven rows per record, with the first row containing an irrelevant search result number, followed a row that only contains the book's title, a row that contains the author (always preceded with the string "by "), a row showing the language the book is written in (always preceded with the string "Language: "), a row showing book's publisher (always preceded with the string "Publisher: ") and, finally, a row containing the string "View all editions and formats" which signifies the last record for a given book.

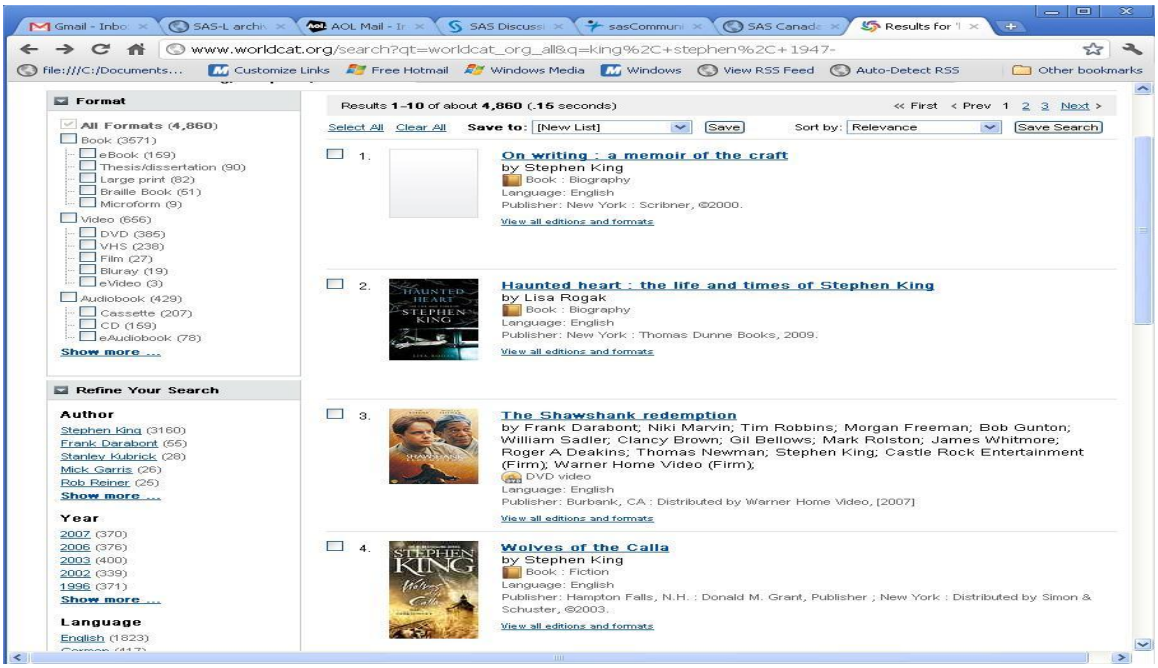


Figure 5
An example of a form or structured printout

The code provided in the present paper can be used to "paste" all four types of tables, as long as the table's structure can be defined. The code includes comments which explain how to specify a number of macro variables that were included to provide mechanisms for accomplishing options which we thought users might appreciate when importing such tables. The present paper describes each of those options, as well as provides suggestions for some freeware that users might find helpful in their efforts to copy and paste data tables from various sources.

THINGS WE'D LIKE TO SEE IN PROC IMPORT

In designing the code for this project we attempted include much of the functionality that PROC IMPORT currently provides for other methods (without, of course, reverse engineering the procedure). However, we discovered some additional functionality that was needed to correctly import the three types of data tables described earlier and noted that some of PROC IMPORT's current options were only included for certain file types. Once we discovered that the original intent of the project turned out to be much easier to accomplish than we had anticipated, we decided to expand the project to investigate additional useful options we would want to include above and beyond those currently offered with PROC IMPORT.

The following options and capabilities are all built into the code provided in the present paper. We could and probably will submit these as SASWare ballot items, but no one would vote for them unless they understood the potential benefits. Our rationale for each is described below.

> **Including the clipbrd as a valid dbms**. While the code offered in this paper provides that capability, it would definitely be preferable to have a routine that was supported, documented, improved with newer versions, and written more efficiently than we could attain using only datastep techniques.

The ability to refer to variables according to their position. This was essential to allow one to name variables that had blank names in a given data table, but it also turned out to be quite useful in adding such things as prefixes, suffixes, formats, informats and measurement units.

The ability to name and rename variables. This was critical when a data table had an unnamed variable, and could always be accomplished in an additional datastep, but we found it to be quite useful as the variable names provided by the original authors were often not the ones we would have chosen.

The ability to add variable name prefixes and suffixes. We discovered a number of cases where the meaning of a variable name was only implied given the context of the web page from which we were obtaining the data. For example, the page might only specify something like 2009, 2010 and 2011 as the variable names for variables 3 thru 5, but really represented actual_2009_revenue, actual_2010_revenue, and actual_2011_revenue.

The ability to specify a map that could be used to parse a structured document. The SAS Institute could probably come up with a better way of defining the map than that which we built into the current code, but such a map is essential in order to import such data.

The ability to add variable labels. Why not?

The ability to capture variables whose names are defined across more than one row, as well as to specify that merged cells should be applied to more than one variable. Currently, PROC IMPORT doesn't do either.

The ability to specify the row at which the data actually begin. PROC IMPORT currently only allows this capability with one file type, but it is applicable to all types of files.

The ability to indicate that data must be transposed. While SAS datasets can always be transposed after they are created, the task is often non-trivial, may require multiple steps and, most often, also requires additional datasteps in order to ensure that the correct variable names, data types, formats and informats have been applied.

The ability to specify which rows should be considered when determining lengths, formats and informats. The current PROC IMPORT GuessingRows' settings vary across file types and are not even available for some data types. More importantly, none of the current options allow one to specify a particular range of data rows that should be used. For example, if the user decides that row six best describes all of the data, they have no direct way to indicate that fact.

The ability to specify variable formats and informats. Sometimes we simply don't want the system to guess, as we know which formats and informats we want applied.

The ability to change any variable's unit of measurement. This option currently isn't available, but we found it to be quite useful.

The ability to assign missing values for specific data. PROC IMPORT currently doesn't provide a way for users to tell the system that certain values, for certain variables, should be considered as missing. Instead, they either have to be dealt with in a subsequent datastep or reflected in numerous undesired log notes.

The ability to indicate whether any specific data should be upcased. Again, why not?

The ability to indicate that certain columns should be dropped. Again, why not?

TRUTH IN ADVERTISING

The code presented in this paper is not intended to be a substitute for PROC IMPORT, may not work on all systems or with all software, should not be used if such use violates any copyright or terms of agreement, is not production quality, and IS ONLY PROVIDED *AS IS* WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. The authors shall not be liable whatsoever for any damages arising out of the use of this documentation or code, including any direct, indirect, or consequential damages. In addition, the authors will provide no support for the materials contained herein.

A number of websites explicitly state that they do not condone extracting data from their pages using the types of methods described in this paper, and some things on the web have similar prohibitions and/or conditions regardless of the methods that might be used to extract the data. Of course, many sites do not have such prohibitions and some only limit the amounts of data that may be extracted. For example, worldcat.org's terms of agreement explicitly prohibits the *use of bots, spiders, or other automated information-gathering devices or programming routines to "mine" or harvest material amounts of Data*. It is the user's responsibility to ensure that harvesting data from any site is a permissible activity.

THE CODE

The code described in this paper, and presented in Appendix I, can be downloaded at:
http://www.sascommunity.org/wiki/Copy_and_Paste_Almost_Anything

DISCLAIMER

The contents of this paper are the work of the authors and do not necessarily represent the opinions, practices or recommendations of their respective organizations.

CONTACT INFORMATION

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

Arthur Tabachneck, Ph.D.
myQNA, Inc.
Thornhill, ON Canada
E-mail: atabachneck@gmail.com

John King
Ouachita Clinical Data Services, Inc.
Mount Ida, AR
E-mail: ouachitaclinicaldataservices@gmail.com

Randy Herbison
Senior Systems Analyst
Westat
1650 Research Boulevard
Rockville, MD 20850
E-mail: RandyHerbison@westat.com

Richard A. DeVenezia
Independent Consultant
9949 East Steuben Road
Remsen, NY13438
http://www.devenezia.com/contact.php

Ben Powell
Genworth Financial
London, England
E-mail: ben.powell@genworth.com

Nate Derby
Stakana Analytics
815 First Avenue, Suite 287
Seattle, WA 98104-1404
E-mail: nderby@stakana.com

APPENDIX I

```
/*********************************I m p o r t a n t*********************************
Name:    paste.sas
Authors: Arthur Tabachneck, John King, Ben Powell, Nate Derby,
         Richard DeVenezia and Randy Herbison
Date:    July 23, 2011
Warnings and Disclaimer: This code is NOT a substitute for PROC IMPORT, may not work
                         on all systems, should NOT be used if such use violates any
                         copyright or terms of agreement, is NOT production quality
                         and is only provided "as is" without warranty of any kind,
                         either express or implied, including, but not limited to, the
                         implied warranties of merchantability, fitness for a
                         particular purpose, or non-infringement.  The authors shall
                         not be liable whatsoever for any damages arising out of the
                         use of this documentation or code, including any direct,
                         indirect, or consequential damages.  In addition, the authors
                         will provide no support for the materials contained herein.
*********************************D i s c l a i m e r*********************************/

options NOQUOTELENMAX;
options datestyle=mdy;
filename clippy clipbrd;
filename revised temp;

%let transpose=NO; *leave as %let transpose=NO;
                   *UNLESS table must be transposed. In such cases set this
                    macro variable to: %let transpose=YES;

%let columns=; *leave as %let columns=;
               *UNLESS table must be transposed or is in long form with only
                one column and each cell represented on a separate row. In such
                cases specify the number of columns the data represent (not the
                number of columns that were copied) e.g., %let columns=4;

%let rows=; *leave as %let rows=;
            *UNLESS table must be transposed or is in long form with only one
             column and each cell represented on a separate row.  In such cases
             specify the number of rows the data represent (not the number of
             rows that were copied), including the rows for both variable names
             and data.  e.g., %let rows=12;

%let look_for=; *leave as %let look_for=;
                *UNLESS the data represent a form rather than a table. If the
                 data represent a form then this macro variable must be set to
                 indicate the string that either begins or ends all data
                 records. E.g., %look_for=View all editions and formats;

%let skip_lines=; *leave as %skip_lines=;
                  *UNLESS the data represent a form rather than a table.  If
                   the data represent a form this macro variable must be set
                   to indicate the number of rows to skip, after the &look_for
                   string has been found, before the first variable's data is
                   found.  E.g., if one blank record separates the &look_for
                   string and the record that contains variable 1, this macro
                   variable would be set as: %skip_lines=1;

%let hrows=1; *indicates that variable names are found on first &hrows. rows.
              A value of 0 indicates that there are no variable names;

%let spaces="      "; *number of consecutive spaces that should be translated
                       to represent a horizontal tab;

%let first_data_row=2; *indicates the row on which the data begin;

%let var_renames=; *specify variables to be named or renamed. A ~ must be used
```

```
                    to separate variable number and variable name, and either a
                    space or different line to specify multiple entries.  E.g.,
                    to specify that variable 1 should be named "Country" and
                    variable 3 should be named "revenue", you would specify:
                    %let var_renames=1~Country
                                     3~revenue;

                   *to indicate that no variables are to be renamed leave the
                    line as: %let var_renames=;

%let var_labels=; *specify any variable labels that you want.  A ~ must be used
                    to separate a variable number and its label, and multiple
                    entries may be on separate line or be separated by spaces.
                    If a label includes embedded spaces, use a ^ to represent
                    each space.  E.g., to specify that variable 2 should be
                    labeled "Street Address and that variable 3 should be
                    labeled "Home Phone", you would specify:
                    %let var_labels=2~Street^Address
                                    3~Home^Phone;

                   *to indicate that you don't want to assign any variable
                    labels leave the line as: %let var_labels=;

%let var_share=; *specify any variables for which a prefix should be taken from
                   another variable's value.

                   For example, if the clipboard contains a table where the
                   string "Revenue" is on the first row but spans across two
                   merged cells, it is likely that the value will only actually
                   exist in the left most cell.

                   Thus, given the following table headers:
                             Revenue     Expenses
                            2010  2011  2010  2011

                   to cause them to be read as: Revenue_2010, Revenue_2011,
                   Expenses_2010 and Expenses_2011 you would specify:

                   %let var_share=3~2
                                  5~4;

                  *to indicate that you don't have any such variable
                   name sharing needs, simply leave the line as:
                   %let var_share=;

%let var_prefix=; *Indicate any string you want added to the left of any
                    variable name. A ~ must be used between variable number(s)
                    and prefixes, and you can include multiple prefixes on
                    either separate lines or separate them with spaces. If you
                    want the same prefix used for a range of variables, specify
                    the range as #-#.  E.g., if variables 2 and 3 are named 1996
                    and 1997, and you want them to be named Price_1996 and
                    Price_1997 you would specify: %let var_prefix=2-3~Price_;

                   *Any variable that starts with a number, and isn't assigned a
                    prefix, will automatically be assigned a prefix of "_". To
                    indicate that no prefixes are to be assigned leave the line
                    as: %let var_prefix=;

%let var_suffix=; *Indicate any string you want added to the right of any
                    variable name. A ~ must be used between variable number(s)
                    and suffixes, and you can include multiple suffixes on
                    either separate lines or separate them with spaces. If you
                    want the same suffix used for a range of variables, specify
                    the range as #-#.  E.g., if variables 2 and 3 are named 1996
                    and 1997, and you want them to be named _1996_cost and
                    _1997_cost, you would specify: %let var_suffix=2-3~_Cost;
```

```
                  *To indicate that no suffixes are to be assigned leave the
                   line as: %let var_suffix=;

%let var_drop=; *leave the line as %let var_drop=; *unless* you want to exclude
                 any variables. Specify any variables that you want dropped
                 from the table.  A ~ must be used to separate variable
                 number(s) and the string "YES", and either a space or separate
                 line to indicate additional entries.  If you want a range of
                 variables to be dropped, specify the range as #-#. E.g., if
                 variable 3 and variables 5 thru 7 are to be dropped, specify:
                 %let var_drop=3~YES
                               5-7~YES;

%let var_upcase=; *leave the line as %let var_upcase=; *unless* you want to
                   upcase any variables. Specify any variables that you want
                   upcased.  A ~ must be used to separate variable number(s)
                   and the string "YES", and either a space or separate line to
                   indicate more than one entry.  If you want to upcase a range
                   of variables specify the range as #-#. E.g., if variable 3
                   and variables 5 thru 7 are to be upcased, specify:
                   %let var_upcase=3~YES
                                   5-7~YES;

%let var_missing=; *specify any values that you want to be considered as
                    missing for any variable.  A ~ must be used to separate
                    variable number(s) and sets of missing values, and either a
                    space or separate line to represent additional entries.  If
                    a missing value includes any embedded spaces, use a ^ to
                    represent each desired space.

                    To specify sets of values, separate each value with an `.

                    E.g., to specify that "n/a" and "n.a." should be considered
                    missing values for variables 2 thru 4, and the number 9
                    considered as missing for variable 5, you would specify:
                    %let var_missing=2-4~n/a`n.a.
                                     5~9;

                   *if don't have any values that you want considered as
                    missing, simply leave the line as: %let var_missing=;
%let var_formats=; *specify any formats that you want applied.  A ~ must be
                    used to separate variable number(s) and formats, and either
                    a space or separate line to represent additional entries.
                    If you want the same format used for a range of variables,
                    specify the range as #-#.  E.g., if you want the format
                    date9. applied for variable 1 and best12. applied for
                    variables 2 thru 4, you would specify:
                    %let var_formats=1~date9.
                                     2-4~best12.;

                   *to indicate that you aren't assigning any formats, leave
                    the line as: %let var_formats=;

%let var_informats=; *specify any informats that you want applied. A ~ must be
                      used to separate variable number(s) and informats, and
                      either a space or separate line to represent additional
                      entries.  If you want the same informat used for a range
                      of variables, specify the range as #-#.  E.g., if you
                      want the informat anydtdte22. to be used for variable 1
                      and best12. to be used for variables 2 thru 4, you would
                      specify:
                      %let var_informats=1~anydtdte22.
                                         2-4~best12.;

                     *to indicate that you don't have any informats to assign
                      leave the line as: %let var_informats=;
```

```
%let var_units=; *specify any number that you want data to be multiplied by.  A
                 ~ must be used to separate variable number(s) and units, and
                 either a space or separate line to represent additional
                 entries.  If you want the same values to be used for a range
                 of variables, specify the range as #-#.

                 E.g., if you want variable 3 to be multiplied by 0.01 and
                 variables 4 thru 7 multiplied by 1,000, you would specify:
                 %let var_units=3~.01
                                4-7~1000;

                *to indicate that you don't have any units to assign leave the
                 line as: %let var_units=;

%let guessingrows=; *specify the range of rows that you want the code to
                     consider in determining formats and informats. Formats and
                     Informats will only be guessed if you do not specify them
                     in the var_formats and var_informats macro variables.

                     E.g., if you only want the third row used to guess the
                     formats and informats, you would specify:
                     %let var_guessingrows=3-3;

                    *to indicate that all rows are to be evaluated, leave the
                     line as: %let var_guessingrows=;

%let outfile=study;

%macro flipfile;
  %if &columns. gt 0 and &rows. gt 0 %then %do;
    %if &transpose. eq YES %then %do;
      data temp;
        infile clippy;
        length temp $32767;
        input;
        _infile_=tranwrd(_infile_, &spaces., '09'x);
        j=_n_;
        do i=1 to &rows.;
          temp=strip(scan(_infile_,i,,"HM"));
          output;
        end;
      run;
    %end;
    %else %do;
      data temp;
        infile clippy;
        length temp $32767;
        input;
        temp=_infile_;
        if _n_ eq 1 then do;
          i=0;
          j=1;
        end;
        i+1;
        output;
        if i eq &rows.+&hrows. then do;
          j+1;
          i=0;
        end;
      run;
    %end;

    proc sort data=temp;
      by i j;
    run;
```

```
  data _null_;
    length holdrec $32767;
    retain holdrec;
    file clippy;
    set temp;
    if mod(_n_,&columns.) eq 1 then holdrec=strip(temp);
    else holdrec=cat(strip(holdrec),"09"x,strip(temp));
    if mod(_n_,&columns.) eq 0 then put holdrec;
  run;

  proc delete data=work.temp;
  run;
%end;
%if "&look_for." ne "" %then %do;
  proc sql noprint;
    select varname into :var_names
      separated by "~"
        from form_varnames
    ;
  quit;

  %let var_cnt=&sqlobs.;
  data _null_;
    infile clippy;
    file revised lrecl=32767;
    length holdrec $32767;
    length temp $32767;
    array varids(&var_cnt.) $32.;
    array varpos(&var_cnt.);
    array varrows(&var_cnt.) $2.;
    array findhead(&var_cnt.);
    retain i varpos varrows varids newrec holdrec findhead;
    input;
    _infile_=tranwrd(_infile_, &spaces., ' ');
    if _n_ eq 1 or newrec eq 1 then do;
      newrec=1;
      if _n_ eq 1 then do;
        /****** obtain and rewrite variable names ******/
        do j=1 to &var_cnt.*5-4 by 5;
          i=input(scan("&var_names.",j,"~"),best12.);
          varids(i)=strip(scan("&var_names.",j+1,"~"));
          if i eq 1 then holdrec=strip(scan("&var_names.",j+1,"~"));
          else holdrec=cat(strip(holdrec),"09"x,strip(scan("&var_names.",j+1,"~")));
          varpos(i)=input(scan("&var_names.",j+2,"~"),3.);
          varrows(i)=scan("&var_names.",j+3,"~");
          findhead(i)=scan("&var_names.",j+4,"~");
          if i eq &var_cnt. then put holdrec;
        end;
      end;

      if _n_ eq 1 or index(_infile_,strip("&look_for.")) gt 0 then do;
        do i=1 to &skip_lines.;
          input;
        end;
        newrec=0;
        i=1;
      end;
    end;

    /****** read and rewrite data ******/
    else do;
      if strip(_infile_) ne "" then do until (doneit);
        doneit=1;
        if index(_infile_,strip(varids(i))) gt 0 or
         findhead(i) eq 0 then do;
          current_first_var=i;
          first_varpos=varpos(i);
```

```
            do j=1 to first_varpos;
              if findhead(current_first_var+j-1) eq 0 then temp=strip(_infile_);
              else do;
                y=index(_infile_,strip(varids(current_first_var+j-1)));
                z=y+length(strip(varids(current_first_var+j-1)));
                temp=strip(substr(_infile_,z));
                if j lt first_varpos then _infile_=substr(_infile_,1,y-1);
                if input(varrows(current_first_var+j-1), ?? best12.)>1 then do;
                  do k= 2 to input(varrows(current_first_var+j-1), best12.);
                    input;
                    temp=catx(" ",temp,strip(_infile_));
                  end;
                end;
                else if strip(varrows(current_first_var+j-1)) eq "U" then do;
                  doingit=1;
                  do while (doingit eq 1);
                    input;
                    if index(_infile_,strip(varids(i+1))) gt 0 then do;
                      doneit=0;
                      doingit=0;
                    end;
                    else if strip(_infile_) ne "" then
                     temp=catx(" ",temp,strip(_infile_));
                    else doingit=0;
                  end;
                end;
              end;
              if i eq 1 then holdrec=strip(temp);
              else holdrec=cat(strip(holdrec),"09"x,strip(temp));
              i+1;
            end;
          end;
          if i gt &var_cnt. then do;
            newrec=1;
            put holdrec;
          end;
        end; /*until doneit*/
      end;
    run;

    data _null_;
      file clippy;
      infile revised lrecl=32767;
      input;
      put _infile_;
    run;
  %end;
%mend flipfile;

%macro expandr (type,string);
  i=1;
  hold_rec="";
  do while (scan("&string.",i," ") ne "");
    if scan(scan(scan("&string.",i," "),1,"~"),2,"-") ne "" then do;
      start=scan(scan(scan("&string.",i," "),1,"~"),1,"-");
      end=scan(scan(scan("&string.",i," "),1,"~"),2,"-");
    end;
    else do;
      start=scan(scan("&string.",i," "),1,"~");
      end=scan(scan("&string.",i," "),1,"~");
    end;
    do j=start to end;
      hold_rec=catx(" ",hold_rec,cat(strip(j)||"~"||
               strip(scan(scan("&string.",i," "),2,"~"))));
    end;
    i+1;
  end;
```

```
  call symput(&type.,strip(hold_rec));
%mend expandr;

%macro filarray (type,string);
  if scan("&string.",i," ") ne "" then
   &type(scan(scan("&string.",i," "),1,"~"))=scan(scan("&string.",i," "),2,"~");
%mend filarray;

/* Note: ***************************************************************************
   the following datastep only needs to be modified if the data represent a form
   rather than a table. If the data represent a form then the following datastep must
   be modified, as described below, to indicate how the data should be read.

   If your data represent a table, then do not modify the following datastep
*************************************************************************************/

 data form_varnames;
  informat varname $50.;
  input varname &;
/* Note: ***************************************************************************
   varname consists of 4 fields separated by a ~.  The fields, from left to right,
   represent: the variable number, the variable name (or, if the variable is preceded
   by a field header, the field header), the field's position within a row (these must
   be declared in right to left order), and the number of rows which have to be read
   to capture the entire field
*************************************************************************************/
  cards;
1~Title~1~1~0
2~by~1~1~1
3~Type~1~1~0
4~Language:~1~1~1
5~Publisher:~1~1~1
;

%flipfile

data _null_;
  length hold_rec $32767;
  infile clippy;
  input;
  _infile_=tranwrd(_infile_, &spaces., '09'x);
  var_count=countc(_infile_,,"H")+1;
  call symput('var_count',strip(put(var_count,8.)));
  %expandr("var_formats",&var_formats.);
  %expandr("var_informats",&var_informats.);
  %expandr("var_missing",&var_missing.);
  %expandr("var_units",&var_units.);
  %expandr("var_prefix",&var_prefix.);
  %expandr("var_suffix",&var_suffix.);
  %expandr("var_upcase",&var_upcase.);
  %expandr("var_drop",&var_drop.);
  %expandr("var_labels",&var_labels.);
  %expandr("var_share",&var_share.);
  stop;
run;
data _null_;
  file revised lrecl=32767;
  infile clippy end=eof;
  array headers(%sysfunc(max(&hrows.,1))) $32767.;
  array varnames(&var_count.) $32.;
  array formats(&var_count.) $32.;
  array informats(&var_count.) $32.;
  array renames(&var_count.) $32.;
  array prefix(&var_count.) $32.;
  array suffix(&var_count.) $32.;
  array labels(&var_count.) $32.;
  array miss(&var_count.) $255.;
```

```
array upcases(&var_count.) $3.;
array drops(&var_count.) $3.;
array units(&var_count.) $32.;
array share(&var_count.) $32.;
array varlens(&var_count.);
array vartypes(&var_count.);
length hold_rec temp ivartype fvartype var_units
       var_names var_labels var_drop $32767;
length missval $255;
retain headers renames varnames vartypes varlens formats informats units prefix
       suffix labels miss upcases drops share grows_start grows_end;
input;
_infile_=tranwrd(_infile_, &spaces., '09'x);
if _n_ le &hrows. then headers(_n_)=tranwrd(tranwrd(tranwrd(
 _infile_, '%', 'percent'),'-','_to_'),'-','_to_');
if _n_ eq &hrows. or (_n_ eq 1 and &hrows eq 0) then do;
  grows_start=scan("&guessingrows.",1,'-');
  if missing(grows_start) then grows_start=&first_data_row.;
  grows_end=scan("&guessingrows.",2,'-');
  if missing(grows_end) then grows_end=999999;
  var_drop="";
  do i=1 to &var_count.;
    %filarray(renames,&var_renames.);
    %filarray(prefix,&var_prefix.);
    %filarray(suffix,&var_suffix.);
    %filarray(units,&var_units.);
    %filarray(formats,&var_formats.);
    %filarray(informats,&var_informats.);
    %filarray(upcases,&var_upcase.);
    %filarray(drops,&var_drop.);
    %filarray(labels,&var_labels.);
    %filarray(miss,&var_missing.);
    %filarray(share,&var_share.);
    if &hrows. eq 0 then varnames(i)=cat("Col"||strip(i));
    else do;
      /****** obtain and assign variable names ******/
      varnames(i)="";
      do j=1 to &hrows.;
        if j eq 1 and share(i) ne "" then do;
          if strip(scan(headers(j),share(i),,"HM")) ne "" then
            varnames(i)=strip(scan(headers(j),share(i),,"HM"));
        end;
        else do;
          if strip(scan(headers(j),i,,"HM")) ne "" then do;
            if strip(varnames(i)) ne "" then varnames(i)=
             strip(varnames(i))||"_"||strip(scan(headers(j),i,,"HM"));
            else varnames(i)=strip(scan(headers(j),i,,"HM"));
          end;
        end;
        if j eq &hrows. and varnames(i) eq "" then varnames(i)=cat("Col"||strip(i));
      end;
    end;
    if renames(i) ne "" then varnames(i)=renames(i);
    if prefix(i) ne "" then varnames(i)=strip(prefix(i))||strip(varnames(i));
    if suffix(i) ne "" then varnames(i)=strip(varnames(i))||strip(suffix(i));
    if strip(labels(i)) eq "" then labels(i)=strip(varnames(i));
    else labels(i)=tranwrd(strip(labels(i)), '^', ' ');
    varnames(i)=tranwrd(strip(varnames(i)),'%', 'percent');
    varnames(i)=tranwrd(strip(varnames(i)),'-','_to_');
    varnames(i)=tranwrd(strip(varnames(i)),'-','_to_');
    varnames(i)=tranwrd(strip(varnames(i)),'#', 'number');
    varnames(i)=tranwrd(strip(varnames(i)), ' ', '_');
    varnames(i)=compress(varnames(i),,'kn');
    if anydigit(substr(varnames(i),1,1)) then varnames(i)=
     cat("_",strip(varnames(i)));
    var_names=catx(" ",var_names,strip(varnames(i)));
    var_labels=cat(strip(var_labels)||"label "||
```

```
     strip(varnames(i))||"="||quote(strip(labels(i)))||";");
    if units(i) ne "" then var_units=catx(" ",var_units,strip(varnames(i))||"="||
     strip(varnames(i))||"*"||strip(units(i))||";");
    if drops(i) eq "YES" then var_drop=catx(" ",var_drop,strip(varnames(i)));
  end;
  if var_drop ne "" then var_drop="(drop="||strip(var_drop)||")";
  call symput('varnames',var_names);
  call symput('varlabls',var_labels);
  call symput('varunits',var_units);
  call symput('vardrop',var_drop);
end;
if _n_ ge  &first_data_row. then do;
  if countc(_infile_,,"H")+1 eq &var_count. then do;
    /****** determine formats and informats ******/
    do i=1 to &var_count.;
      temp=strip(scan(_infile_,i,,"HM"));
      if upcase(upcases(i)) eq "YES" then temp=upcase(temp);
      if strip(temp) ne "" then do;
        if miss(i) ne "" then do;
          k=1;
          do while (scan(miss(i),k,"` ") ne "");
            missval=tranwrd(strip(scan(miss(i),k,"` ")),'^',' ');
            temp=tranwrd(strip(temp),strip(missval), '');
            k+1;
          end;
        end;
        if grows_start LE _n_ and grows_end GE _n_ then do;
          call missing(vartype);
          in_test = input(temp, ?? best12.);
          if not missing(in_test) then vartype=0;
          else do;
            in_test = input(temp, ?? anydtdte21.);
            if not missing(in_test) then vartype=2;
            else do;
              if index(temp,"$") then in_test = input(temp, ?? dollar21.);
              if not missing(in_test) then vartype=4;
              else do;
                if index(temp,",") then in_test = input(temp, ?? comma21.);
                if not missing(in_test) then vartype=5;
                else do;
                  if index(temp,"%") then in_test=input(temp, ?? percent21.);
                  if not missing(in_test) then vartype=3;
                  else vartype=1;
                end;
              end;
            end;
          end;
          if missing(vartypes(i)) then vartypes(i)=vartype;
          else if vartype ne vartypes(i) then vartypes(i)=1;
          if missing(varlens(i)) or length(temp) gt varlens(i) then
           varlens(i)=length(temp);
        end;
      end;
      if i eq 1 then hold_rec=strip(temp);
      else hold_rec=cat(strip(hold_rec),"09"x,strip(temp));
    end;
    put hold_rec;
  end;

  /****** assign formats and informats ******/
  if eof then do;
    ivartype="";
    fvartype="";
    do i=1 to &var_count.;
      if vartypes(i)=1 then do;
        itempvar=cat("$",strip(put(varlens(i),3.)),".");
        ftempvar=itempvar;
```

```
        end;
        else if vartypes(i)=2 then do;
          itempvar="anydtdte.";
          ftempvar="date9.";
        end;
        else if vartypes(i)=3 then do;
          itempvar="percent.";
          ftempvar="percent8.2";
        end;
        else if vartypes(i)=4 then do;
          itempvar="dollar.";
          ftempvar=cat("dollar",strip(put(varlens(i),3.)),".");
        end;
        else if vartypes(i)=5 then do;
          itempvar="comma.";
          ftempvar=cat("comma",strip(put(varlens(i),3.)),".");
        end;
        else do;
          itempvar="best12.";
          ftempvar="best12.";
        end;
        if strip(informats(i)) ne "" then itempvar=strip(informats(i));
        if strip(formats(i)) ne "" then ftempvar=strip(formats(i));
        ivartype=catx(" ",ivartype,"informat",varnames(i),itempvar,";");
        fvartype=catx(" ",fvartype,"format",varnames(i),ftempvar,";");
      end;
      call symput('informt',ivartype);
      call symput('formt',fvartype);
    end;
  end;
run;

options QUOTELENMAX;

data &outfile. &vardrop.;
  infile revised lrecl=32767 dsd delimiter="09"x;
  &informt.;
  &formt.;
  &varlabls.;
  input &varnames.;
  &varunits.;
run;

proc delete data=work.form_varnames;
run;

filename clippy clear;
filename revised clear;
```