PROC QPRINT: THE FAST AND EASY WAY TO COMPLEX LISTINGS (OR HOW TO AVOID DATA __NULL__S)

George DeMuth, Burroughs Wellcome Co.
Cam Huffman, Burroughs Wellcome Co.

ABSTRACT

When generating reports, one must often create listings and tables which require complex or tailored specifications. The means to achieve such a listing may necessitate a more adaptable SAS facility than PROC PRINT. In such cases, DATA __NULL__s are often used to customize listings and can produce impressive results. However, DATA __NULL__s also require intricate work when minor modifications are necessary, such as adding another variable to the listing or changing a heading.

An alternative SAS facility that combines the sophistication of the DATA __NULL__ with the "easy to use" features of PROC PRINT is PROC QPRINT.

PROC QPRINT is similar to PROC PRINT in syntax, but gives you greater control of the listing format. A disadvantage of using PROC QPRINT in customized listings is its inability to duplicate the action of certain options and features in PROC PRINT and DATA __NULL__. Fortunately, these drawbacks can easily be overcome by using a short macro developed specifically to be used in conjunction with PROC QPRINT....QSORT.

INTRODUCTION

The Clinical Statistics department of the Burroughs Wellcome Co. generates statistical reports containing summary tables, graphic figures, and data listings which present safety and efficacy analyses of pharmaceutical products. The reports are used within the company and eventually included in a New Drug Application (NDA) prepared for the US Food and Drug Administration (FDA). Commonly, many studies are performed on a single product, and these reports often have a similar format.

As a result, it is clear that an efficient way to generate a series of similar reports needed for one drug product is to share programs between studies. This simple idea usually saves much time and reduces repetitive programming. The next step to improve efficiency is to make production of listing programs faster and easier.

PROBLEM

In the past, practically all of the general listings and summaries included in the study reports were generated by using DATA __NULL__s and PROC PRINTS. There are problems with these SAS facilities when generating listings and summaries. PROC PRINT cannot produce tailored or complex tables and listings that many statisticians and reviewers want. Therefore, DATA __NULL__s are used for this purpose. However, while DATA __NULL__s are capable of producing complex tables and listings, DATA __NULL__s can be tedious and time consuming to program.

In the context of sharing programs for use in other similar reports, DATA __NULL__s can also require intricate work when minor modifications are necessary, such as adding another variable to a listing or changing a heading. This fact reduces the efficiency gained when programming tasks are shared between similar studies.

SOLUTION

What was needed was a programming facility that could produce detailed and customized listings and that would be easy to modify for sharing purposes. The partial solution was PROC QPRINT. By reading the SAS Technical Report P-145 (Changes and Enhancements to the Version 5 SAS System, April 1986) and exploring the many powerful and interchangeable commands available in PROC QPRINT we felt we had found just what was needed....almost.

PROC QPRINT can be used in place of DATA __NULL__s in most listings and tables and uses PROC PRINT-style syntax. However, a disadvantage of using PROC QPRINT is its inability to duplicate certain actions available in PROC PRINT and in the DATA __NULL__. Specifically, this is the "BY variable; ID variable" option in PROC PRINT, or the "FIRST.variable" facility in the DATA __NULL__.

Our solution to this dilemma is a short macro named QSORT. It was developed specifically to be used in conjunction with PROC QPRINT. By using the PROC QPRINT/QSORT combination, one can easily and efficiently produce complicated listings with the simplicity of a PROC PRINT and without the drawbacks of a DATA __NULL__.

Next, we will introduce PROC QPRINT to those who are not familiar with it. The basics of PROC QPRINT are covered (distilled from the SAS technical report) along with several examples of what PROC QPRINT will do on its own. Following that section, the QSORT macro is introduced and explained. Guidelines for how to use it and examples of what it can do in conjunction with PROC QPRINT are also presented.

USING PROC QPRINT

The QPRINT procedure prints the values in a SAS data set. PROC QPRINT has an extensive list of commands that give the user a large degree of control of the header area of the listing. For instance, PROC QPRINT can center titles and create titles that span several variable columns.

A brief outline and description of the PROC QPRINT statements the authors use follows. This is a good place to start if the user has not worked with the QPRINT procedure before. If PROC QPRINT is capable of easing the reader's programming tasks, then we recommend acquiring the Technical Manual which includes documentation of this procedure (see references).

The most commonly used QPRINT statements are:

PROC QPRINT	the procedure call can be used to declare options that apply to all VAR and ID statements.
OBS	option to print column with the observation numbers.
ID/VAR	specify variables to be printed and sets options specific to printing those variables.

HEAD / TAIL	used to attach headers that span multiple columns.
FOOT / TAIL	used to attach footers that span multiple columns.
CONSTANT	prints a constant in each line of output.

Also, FORMAT, TITLE, FOOTNOTE, BY, and LABEL statements are used in the usual manner with QPRINT.

The next section describes with brief examples what QPRINT does, then several complete examples are shown.

PROC statement

The PROC statement can be used to specify options for all ID, CONSTANT, and VAR statements. For example, JUSTIFY=CENTER center justifies all the the column headers. The statement we usually use is

```
PROC QPRINT UNDERLINE='_'
  BLANKS(HEAD=1) HEAD=LABEL
  JUSTIFY=CENTER ;
```

UNDERLINE='_' creates a solid underline between the header and the data.

BLANKS(HEAD=1) causes 1 blank line to appear between the underline and the first line of data. This makes the underline evenly spaced between the header and the data.

JUSTIFY=CENTER makes the all the headers be center justified. (default justifies like PROC PRINT).

HEAD=LABEL causes the labels to be used as column headers.

ID statement

Used for specifying ID variables to appear on the left of all listing pages.

VAR statement

Used to specify listing variables and options to be used with those variables.

ID statements must appear before VAR statements. You can have as many of either as you wish. The options available to ID and VAR are the same and are used to describe how the variables are to be printed.

Options are placed after a '/' character in the ID or VAR statement.

A string in quotes in the option area will span all the variables in a given ID or VAR statement. An example VAR statement is:

```
VAR SYST DIAS/'BLOOD PRESSURE' F=3.;
```

This would print SYST and DIAS, each with format of 3., and cause 'BLOOD PRESSURE' to span both columns. Note, we also could have said FORMAT=3. in the options area. QPRINT accepts a number of abbreviations like this.

It is very common to change how the variable labels are justified by using the option area of the VAR statement. J=L, J=C, and J=R give left, center, and right justified column headers, respectively.

HEAD / TAIL

Each HEAD statement is matched with a TAIL and is used to center titles over the variables in multiple VAR or ID statements. The HEAD statement specifies the string, while the TAIL tells PROC QPRINT how many variables the string is to be centered over. This is similar to the effect of the 'quotedstring' in the ID or VAR statements.

An example is:

```
HEAD 'Location' LINE=1 ;
  HEAD '/-/' XPAND LINE=1 ;
    VAR STATE / F=$5. ;
    VAR CITY / F=$5. ;
  TAIL ;
TAIL ;
```

which would produce a title statement that looked like:

```
  Location
 ----------
STATE  CITY
```

In this example, XPAND or X causes the '-' to cover the length of the columns. XPAND will expand the first and last character of any string to the end of the field that string spans.

The LINE option causes the headers to appear 1 line above the variables they are spanning. The default for HEAD/TAIL strings is to appear 2 lines above the variables they span.

FOOT / TAIL

FOOT acts like the HEAD statement, but puts a footer on a column.

CONSTANT

The constant statement allows you to quote a string that will be printed for every record. For instance, it could be used to add a '(' and ')' to a listing.

LABELs

Labels can be multi-level; similar to PROC PRINT, but are more flexible. More than 3 levels are allowed and the labels are stacked rather than put on the highest row, the opposite of PROC PRINT.

PROC QPRINT uses delimiters in variable labels to decide how a label is printed. Each delimiter in a label causes a new level to be printed in the column header. Instead of the SPLIT= option required by PROC PRINT, PROC QPRINT looks at the first character in a label. If first character is not A-Z or 1-0, it is used as the delimiter for that label.

An example is:

```
LABEL PULSE = '#Heart
      Rate#(Beats/#Minute)#';
```

Left justified, this label would appear as:

```
Heart Rate
(Beats /
Minute)
```

The first character in the label is '#', this is not a letter or a numeric value, it is used as a delimiter for the label.

Note, for very long column headers the user may desire to use more than the 40 characters (including delimiters) allowed in a label. One easy way to do this is to specify part of the column header with HEAD/TAIL statements.

Other SAS

The other SAS statements used with PROC QPRINT are TITLE, FOOTNOTE, and BY statements. TITLE and FOOTNOTE statements are used as always.

The BY statement will page when a new BY group is hit and the BY group will appear in the title. FORMAT and LABEL statements must be used to control the format of the BY variable.

Example PROC QPRINT code with output

Following is an example of QPRINT with the output that the code would produce. This is a typical example of the types of output that can easily be produced with PROC QPRINT.

EXAMPLE 1 -- Simple QPRINT procedure

Example code:

```
OPTIONS LINESIZE=80 ;

PROC QPRINT UNDERLINE='_' BLANKS(HEAD=1) HEAD=LABEL
   JUSTIFY=CENTER ;
  VAR NAMES / F=$20. J=L ;
  VAR AGE YN / '/-AGE QUESTIONS-/' X ;
  VAR TIME / F=5.1 ;
  LABEL NAMES = 'NAME'
     AGE = '/PERSON''S/AGE/'
    YN = '/OVER/45/YEARS/OF/AGE/'
    TIME = '/TIME TO/RETIREMENT/' ;
  TITLE 'TEST OF QPRINT PROCEDURE' ;
```

Example output:

TEST OF QPRINT PROCEDURE

NAME	---AGE QUESTIONS-- PERSON'S AGE	OVER 45 YEARS OF AGE	TIME TO RETIREMENT
GEORGE	86.4421	YES	13.6
PAUL	74.3684	YES	25.6
ELVIS	6.07588	NO	93.9
KONG	39.9579	NO	60.0
KIRK	23.0000	NO	77.0
OTIS	80.0000	YES	20.0

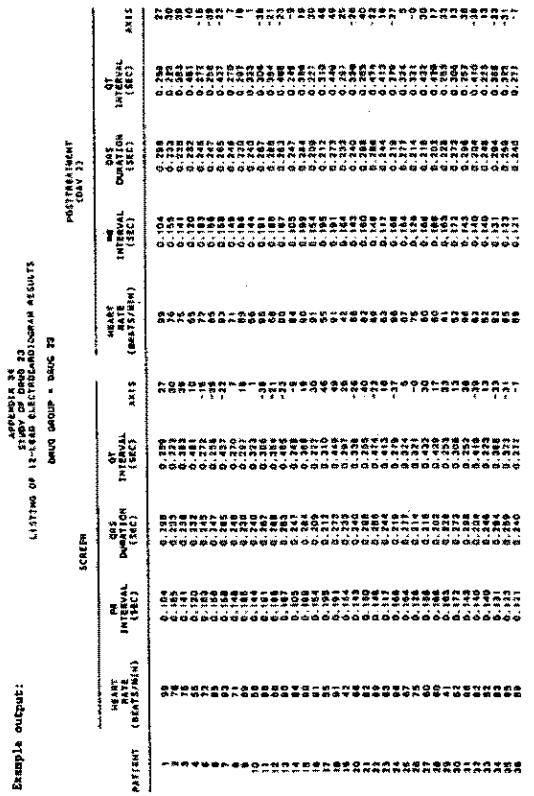
EXAMPLE 2 -- Extended QPRINT procedure

Example code:

```
OPTIONS NODATE NONUMBER PS=55 PS=55 ;

PROC QPRINT DATA=EKG UNDERLINE='_' BLANKS(HEAD=1) HEAD=LABEL
     JUSTIFY=CENTER ;
  BY GROUP ;
  ID PT / F=3. ;
  HEAD 'SCREEN' LINE=1 ;
    HEAD '/_/' X ;
      VAR SHEART / F=3. ;
      VAR SPR SQRS SQT / F=5.3 ;
      VAR SAXIS / F=3. ;
    TAIL ;
  TAIL ;
  HEAD 'POSTTREATMENT' LINE=1 ;
    HEAD (DAY 2) ' LINE=1 ;
      HEAD '/_/' X ;
        VAR PHEART / F=3. ;
        VAR PPR PQRS PQT / F=5.3 ;
        VAR PAXIS / F=3. ;
      TAIL ;
    TAIL ;
  TAIL ;
  TITLE ' ' ;
  TITLE5 'APPENDIX 34' ;
  TITLE6 'STUDY OF DRUG 23' ;
  TITLE7 'LISTING OF 12-LEAD ELECTROCARDIOGRAM RESULTS' ;
  FORMAT GROUP $GRPF. ;
  LABEL GROUP='DRUG GROUP'
        PT = 'PATIENT'
        SHEART = '#HEART#RATE#(BEATS/MIN)#'
        SPR = '#PR#INTERVAL#(SEC)#'
        SQRS = '#QRS#DURATION#(SEC)#'
        SQT = '#QT#INTERVAL#(SEC)#'
        PHEART = '#HEART#RATE#(BEATS/MIN)#'
        PPR = '#PR#INTERVAL#(SEC)#'
        PQRS = '#QRS#DURATION#(SEC)#'
        PQT = '#QT#INTERVAL#(SEC)#'
        SAXIS = 'AXIS'
        PAXIS = 'AXIS' ;
```

Example output:



USING THE QSORT MACRO

This section describes the input required by the QSORT macro and where QSORT should be used in a SAS program. An example is provided with output, so that the reader can examine the type of output produced.

Calling the QSORT macro

The macro call is:

```
%QSORT  (data set name, format data set name, list of sorting
         variables, list of other variables);
```

The first field in the macro call is the name of the data set which is to be listed. The second field is the format data set, described below, which contains the formats of the variables to be listed. The third field lists all variables which should appear in the BY/ID style, in the order in which they should be sorted. The last field should contain the names of all other variables that appear in the listing.

Variables that are used for paging are BY variables in PROC QPRINT, they should not appear in the macro call.

The format data set is used by the macro to obtain formats to assign to the variables in the sorting and listing fields. CALL SYMPUT is used in a DATA _NULL_ to create macro variables with the formats. The format data set can have any name, but it must contain two character fields called VBLE and FORMAT. The VBLE variable contains the name of a variable, the FORMAT variable contains the format the variable is to be printed in.

The output of QSORT is a data set called QSET, this contains the formatted values of the variables ready for PROC QPRINT. If paging by a group is desired, sort after the macro call.

Debugging Tips

The authors have seen only a few errors with the QSORT macro. The first involves incorrect or missing formats in the format data set. The code will fail if an incorrect format is used or the format for a sorting or listing variable is missing. The second error occurs when a macro variable is reset, because the QSORT macro is nested in another macro call. This does not effect the QSORT macro, but can effect your other macro and can be fixed by renaming one of the macro variables. Lastly, listing and sorting variables that are not in the data set will either result in an error or uninitialized variable.

Example code for QSORT

Ex. 1) Example format data set:

```
DATA FORMSET ;
  INPUT VBLE $ FORMAT $ ;
  CARDS ;
PATIENT 3.
DAY     DAYF.
SYST    3.
DIAS    3.
PULSE   3.
DATE    MMDDYY8.
...
        ;

%QSORT(DATASET,FORMSET,PATIENT DAY,SYST DIAS
PULSE DATE ...);
```

The resulting QSET data set would by sorted by PATIENT and PHASE and all variables would be formatted as specified in the format data set.

Ex. 2) Getting paging with a BY statement:

```
%QSORT(DATASET,FORMSET,PT PHASE,SYST DIAS
PULSE DATE ...);

. PROC SORT DATA=QSET ;
   BY GROUP ;

PROC QPRINT DATA=QSET UNDERLINE='__'
  BLANKS(HEAD=1) HEAD=LABEL
  JUSTIFY=CENTER ;
  BY GROUP ;
  FORMAT GROUP $GRPF. ;
  ID PT PHASE ;
  ....
```

The resulting listing would have paging on the first appearance of a new GROUP value. Note that GROUP must be formatted in the QPRINT procedure with a standard FORMAT statement.

Example QSORT code with output

Following is an example of QSORT and PROC QPRINT with the output that the code would produce. This is a typical example of the types of output that can easily be produced with PROC QPRINT.

EXAMPLE 3 - QSORT/QPRINT procedure

Example code:

```
*************************************************************************
* CREATE FORMAT DATA SET AND PROCESS DATA WITH QSORT MACRO
*************************************************************************

%INCLUDE QSORT / NOSOURCE2 ;

DATA FORMSET ;
   INPUT VBLE $ FORMAT $ ;
   CARDS ;
PATIENT      2.
DAY          DAYF.
HOUR         1.
SYST         3.
DIAS         3.
PULSE        3.
BLUSH        BLUSHF.
   ;

%QSORT(FAKE,FORMSET,PATIENT DAY,HOUR SYST DIAS PULSE BLUSH);

PROC SORT DATA=QSET ;
   BY GROUP ;

OPTIONS NODATE NONUMBER LINESIZE=80 PS=55 TPS=55 ;

PROC QPRINT  UNDERLINE='_' BLANKS(HEAD=1) HEAD=LABEL JUSTIFY=CENTER ;
   BY GROUP ;
   ID PATIENT DAY ;
   VAR HOUR ;
   VAR SYST DIAS / '/ BLOOD PRESSURE / (MM HG) /-/' K L=1 ;
   VAR PULSE ;
   VAR BLUSH / J=L ;
   FORMAT GROUP $GRPF. ;
   TITLE ' ' ;
   TITLE4 'APPENDIX 23' ;
   TITLE6 'STUDY OF DRUG 23' ;
   TITLE8 'LISTING OF CARDIOVASCULAR DATA' ;
   LABEL GROUP = 'DRUG GROUP'
         DAY = '/TEST/DAY/'
         SYST = 'SYSTOLIC'
         DIAS = 'DIASTOLIC'
         PULSE = '#HEART RATE#(BEATS /#MINUTE)#'
         BLUSH = '/FLUSHING/PRESENT?/' ;
```

Example output:

APPENDIX 23

STUDY OF DRUG 23

LISTING OF CARDIOVASCULAR DATA

DRUG GROUP = PLACEBO

PATIENT	TEST DAY	HOUR	BLOOD PRESSURE (MM HG) SYSTOLIC	BLOOD PRESSURE (MM HG) DIASTOLIC	HEART RATE (BEATS / MINUTE)	FLUSHING PRESENT?
1	DAY 1	1	106	77	57	NO FLUSHING
		2	137	89	99	NO FLUSHING
	DAY 15	1	169	69	71	NO FLUSHING
	DAY 30	1	117	92	81	NO FLUSHING
2	DAY 1	1	151	108	55	NO FLUSHING
		2	126	99	73	NO FLUSHING
	DAY 15	1	168	97	78	NO FLUSHING
	DAY 30	1	136	64	90	FLUSHING
3	DAY 1	1	164	63	70	FLUSHING
		2	144	69	58	FLUSHING
	DAY 15	1	178	98	52	NO FLUSHING
	DAY 30	1	133	73	90	NO FLUSHING
4	DAY 1	1	170	85	68	NO FLUSHING
		2	130	78	51	NO FLUSHING
	DAY 15	1	113	89	78	FLUSHING
	DAY 30	1	178	93	87	FLUSHING
5	DAY 1	1	174	88	92	FLUSHING
		2	131	60	58	FLUSHING
	DAY 15	1	110	86	95	FLUSHING
	DAY 30	1	147	79	57	NO FLUSHING
6	DAY 1	1	112	87	81	FLUSHING
		2	178	99	99	FLUSHING
	DAY 15	1	157	95	56	NO FLUSHING
	DAY 30	1	111	84	90	NO FLUSHING
7	DAY 1	1	139	63	64	NO FLUSHING
		2	145	93	82	NO FLUSHING
	DAY 15	1	118	83	42	NO FLUSHING
	DAY 30	1	148	97	70	NO FLUSHING
8	DAY 1	1	177	95	83	NO FLUSHING
		2	103	79	65	NO FLUSHING
	DAY 15	1	138	81	63	NO FLUSHING
	DAY 30	1	149	68	76	NO FLUSHING

DESCRIPTION OF QSORT MACRO

The QSORT macro creates a data set to be used with PROC QPRINT. The new data set, called QSET, contains formatted data in the BY/ID style achieved with PROC PRINT. The algorithm the macro uses is described briefly below. We suggest that you examine the code and it's documentation for a more thorough review.

QSORT algorithm:

1. Scan the lists of sorting variables and listing variables to create macro variables with substitute names. These substitute variables will receive the formatted data values.
2. Print a list of substitute names and the variables they represent in the SASLOG.
3. Use CALL SYMPUT to load formats in the format data set into macro variables.
4. Sort the data with the sorting variables list.
5. Create the output data set by converting the original data into character data with the formats specified in the format data set. The following rules are used to create the data set:
   a. if a variable is a sorting variable, it is printed only on the first occurrence. (FIRST.variable).
   b. listing variables are printed for every record.
   c. a blank line is inserted after the last value of the first sort variable is reached. (LAST.variable).
6. Rename the substitute variables to the original variable names.

The QSORT macro presented below can be used as a starting point for more complex versions. With fairly little work, macros have been created that print the values of the sorting variables when a new page is reached and allow a blocking variable to create extra blank lines. We also have a macro to create variable format data sets from PROC CONTENTS information.

Listing of the QSORT macro code

The following is a listing of the QSORT macro source code:

```
*************************************************
** QSORT macro is designed for use with PROC QPRINT
** to create data null type listings.
** It outputs a data set called QSET to be used
** with PROC QPRINT using the same variable names
** specified in the macro call.  The variables are
** converted to character using the formats in the
** VLBES data set.
*************************************************;

*************************************************
** Macro variables:
**
**    DATASET  = Name of the input data set
**    VBLES    = Name of the VBLES data set
**                 w/variable formats
**    BYVARS   = List of sorting variables
**    LVARS    = List of variable to be listing
*************************************************;

OPTIONS DQUOTE ;

%MACRO QSORT(DATASET,VBLES,BYVARS,LVARS);

 *-----------------------------------------------*
 ¦Create list of macro variables containing the
 ¦names of the BY variables.  Also, create new
 ¦variable names to hold the character values.
 *-----------------------------------------------;

 %LET NBY = 0 ;
 %LET BVAR = %SCAN(&BYVARS,1);

 %DO %WHILE(&BVAR ^= ) ;
   %LET NBY = %EVAL(&NBY + 1);
   %LET BVAR&NBY = &BVAR ;
   %LET NEWBY&NBY = ___BYV&NBY ;
   %LET BVAR = %SCAN(&BYVARS,%EVAL(&NBY + 1)) ;
 %END;

 *-----------------------------------------------*
 ¦ Create list of macro variables containing the
 ¦ names of the list variables and the character
 ¦ substitutes.
 *-----------------------------------------------;

 %LET NL = 0 ;
 %LET LVAR = %SCAN(&LVARS,1);

 %DO %WHILE(&LVAR  .= ) ;
   %LET NL = %EVAL(&NL + 1);
   %LET LVAR&NL = &LVAR ;
   %LET NEWL&NL = ___LV&NL ;
   %LET LVAR = %SCAN(&LVARS,%EVAL(&NL + 1)) ;
 %END;
```

```
*-------------------------------------------------*
| Print list of BY and listing variables in the   |
| saslog as a check. (optional code).             |
*-------------------------------------------------;

%PUT ;
%PUT LIST OF INTERMEDIATE VARIABLES ;
%PUT LISTING OF BY VARIABLES ;
%DO I=1 %TO &NBY ;
  %PUT &&NEWBY&I = &&BVAR&I ;
%END ;
%PUT  ;
%PUT LISTING OF LIST VARIABLES ;
%DO I=1 %TO &NL ;
  %PUT &&NEWL&I = &&LVAR&I ;
%END ;
%PUT ;

*-------------------------------------------------*
| Get variable formats from the VBLES data set as |
| macro variables (BYFn and LFn). Use these       |
| formats to create the ____BYV and ____LV vari-  |
| ables in the QSET data set.                     |
*-------------------------------------------------;

DATA _NULL_ ;
  SET &VBLES ;
  %DO I=1 %TO &NBY ;
    IF VBLE = "&&BVAR&I" THEN CALL
    SYMPUT("BYF&I",PUT(FORMAT,$10.));
  %END ;
  %DO I=1 %TO &NL ;
    IF VBLE = "&&LVAR&I" THEN CALL
    SYMPUT("LF&I",PUT(FORMAT,$10.));
  %END ;

*-------------------------------------------------*
| Sort the data set by the variables in the BY    |
| list.  Create a new data set (QSET) with BY/ID  |
| format of sort variables and blank lines        |
| inserted.  This data set is used w/PROC QPRINT. |
*-------------------------------------------------;

  PROC SORT DATA=&DATASET ;
    BY &BYVARS ;

  DATA QSET ;
    SET &DATASET END=EOF;
    BY &BYVARS ;
    **** For each BY variable, keep if
    **** FIRST.variable, else insert a blank.  ;
    %DO I=1 %TO &NBY ;
    IF FIRST.&&BVAR&I THEN &&NEWBY&I =
    PUT(&&BVAR&I,&&BYF&I);
      ELSE &&NEWBY&I = ' ' ;
    %END ;
    %DO I=1 %TO &NL ;
      &&NEWL&I = PUT(&&LVAR&I,&&LF&I);
    %END ;
    OUTPUT ;
  IF LAST.&BVAR1 AND EOF=0 THEN DO;
   .%DO I=1 %TO &NBY ;
      &&NEWBY&I = ' ';
    %END ;
    %DO I=1 %TO &NL ;

      &&NEWL&I = ' ' ;
    %END;
    OUTPUT ;
  END;

*-------------------------------------------------*
| Rename all the ____BYV and ____LV variables to  |
| the names of the original variables.            |
*-------------------------------------------------;

DATA QSET  ;
  SET QSET  ;
  DROP %DO I=1 %TO &NBY ;
    &&BVAR&I
    %END ;
    %DO I=1 %TO &NL ;
    &&LVAR&I
    %END ; ;

DATA QSET ;
  SET QSET  ;
  RENAME %DO I=1 %TO &NBY ;
       &&NEWBY&I = &&BVAR&I
    %END ;
    %DO I=1 %TO &NL ;
      &&NEWL&I = &&LVAR&I
    %END ; ;

  RUN;

%MEND QSORT ;
```

REFERENCES

The QPRINT procedure is documented in the SAS Technical Report P-145 (Changes and Enhancements to the Version 5 SAS System, April 1986) pages 89-122.

The authors may be reached at:
Burroughs Wellcome Co.
3030 Cornwallis Road
Research Triangle Park, NC 27709