ABSTRACT
Programmers often need to map data to an alternative structure to satisfy the specifications and requirements of their clients. Nearly half the effort devoted to data restructuring and mapping is spent ensuring that variables have necessary attributes (lengths, labels, formats) and structure (row order, column order). While it is important to maintain a desired structure, the time that is devoted to these tasks may be better spent insuring all data is included and that proper mapping from one form to another is preserved.

A method is presented that uses SAS code to generate the necessary structure and attributes from client provided specifications using a few PROC SQL statements and macro processing. Accessing specifications directly, using the SELECT and INTO clauses from PROC SQL, will create macro variables that can be processed such that all attributes and structure composition statements are automatically created.

Utilizing the MPRINT and MFILE options, a new program to be applied where necessary may be created. Very often, if one set of specifications is used for a project, chances are they will be used on a subsequent project. By using these options, the new program has all the macro variables created for the previous project resolved within the program. This provides an easier to read utility for programmers that inherit the responsibility of data customization but may not be as comfortable using SQL or macros.

INTRODUCTION
Clinical SAS programmers are increasingly required to customize data used for reporting results of pharmacokinetic studies. While the studies are conducted and the data collected according to one standard, for reporting or regulatory submission, an alternative format may be necessary, often dictated by mandated regulatory standards (CDISC) or internal client warehousing preferences. Nearly all the data collected by CROs (Clinical Research Organizations) while conducting pharmacological studies require restructuring according to a client-defined specification at the conclusion of investigation.

Unfortunately, much of the time is spent focusing on cosmetic details such as variable names and labels (general attributes) and dataset composition (such as sorting order and column placement). While correctly spelling a variable, or attaching the correct label, or insuring proper column placement may be important for submission, data analysts must focus most of their attention to properly maintaining data integrity throughout the mapping process.

When electronic data specifications are available, either in spreadsheet or text format, SAS programmers have the option of exploiting these documents to directly initialize dataset variables and structure. The process presented here relies on the use of PROC SQL to generate several macro variables. Once created, an attribute statement within a %DO loop initializes the necessary dataset variables. Also discussed is the usefulness of the MFILE and MPRINT options when coupled with this approach.

PROCESS AND CODING (please refer to Figure 1 throughout)
The availability of specifications, either client provided or from a previous project, is necessary to begin the process of automation. The first step of the process is the creation of a permanent SAS dataset to act as a repository for variables, their attributes, and any information pertaining to dataset composition. Alternatively, if a previous project were conducted with the same specifications, direct access to output from the CONTENTS or the DATASETS procedure would certainly expedite the process.

In Figure 2, an example of a permanent SAS dataset (name STRUCTURE) created from available specifications is shown. According to this example, four datasets, are required (AES, DEM, MED, VIT) for the final data submission. Within each panel are necessary VARIABLES and various attributes (LABEL, LENGTH), column order (RETORD), and a label for the dataset itself (DDESC). The process will be illustrated with the DEM panel subset.

Using PROC SQL, specifically the SELECT and INTO clause, several macro variables are created which may be used to establish the new dataset. The first step to automating the initialization of variable attributes is to determine the number of variables that will be required. This can be done with the following line of code within SQL:

```
select n(variable) into :nvar from permanent.structure where panel="DEM";
```
Using this statement, the number of variables in the DEM panel (in this case 4) are counted and placed into the macro variable &nvar. The macro variable &nvar will be important later in the process when only the necessary number of variables required for the final submission are initialized and kept.

Next, the four variables (BIRTHDAT, AGE, RACE, SEX) will be assigned to &var1 - &var4 respectively, and similarly for the labels and lengths:

```
select variable into :var1-:var%left(&nvar)
from permanent.structure where panel="DEM";
select description into :label1-:label%left(&nvar)
from permanent.structure where panel="DEM";
select length into :length1-:length%left(&nvar)
from permanent.structure where panel="DEM";
```

In the first statement, the VARIABLE column in the STRUCTURE dataset will be accessed, and then macro variables &var1 - &var4 will be set to BIRTHDAT, AGE, RACE, and SEX, respectively. Similarly, &label1 - &label4 will be created (BIRTH DATE, AGE (YRS), RACE, and GENDER), as is &length1 - &length4 ($20., 3., 3., and 3.).

Since column order within delivered datasets is often important, the use of the RETORD variable in the STRUCTURE dataset will allow us to exploit the order in which the client has provided the specifications, and will also allow us to create a variable list of what needs to be kept within the final dataset. By maintaining the order of RETORD, again using an SQL statement with the SELECT and INTO clauses, &retain will be equal to the string BIRTHDAT AGE RACE SEX,

```
select variable into :retain separated by ' '
from permanent.structure where panel="DEM";
```

In this example, the SEPERATED BY clause within PROC SQL is also used. The use of an empty space was necessary to create the character string desired (for eventual use as SAS code). Other characters may be used depending on desired application.

The last macro variable created will be used to hold the name, or label, that must be attached to the final dataset. In this case, the variable &describe will be DEMOGRAPHY:

```
select ddesc into :describe
from permanent.structure where panel="DEM";
```

All the necessary information to initialize variables and establish the composition of the dataset is now available - all from six lines of code within the SQL procedure. Using the created macro variables, the new dataset structure is initialized automatically.

To establish the variables, all the necessary ATTRIBUTE statements are placed within the %DO loop:

```
%macro NEWDATA;

data NEW;
%do i=1 %to &nvar.;
    attrib &&var&i.. label"&&label&i.. length=&&length&i..;
%end;
;
set Original.SAS_DATA;
*** Mapping ***;
run;
%mend NEWDATA;

%NEWDATA;
```

Following the establishment of the NEW dataset with the variables and their attributes, the original dataset is used as a starting point. The necessary logic to transform the actual data from the old form to the new form is referred to as Mapping (Figure 1).
Once all necessary mapping and transformations have been completed, a process which will, and should, take more time and careful attention, a final dataset is established, utilizing the &label and &retain generated from the STRUCTURE dataset established from the original specifications available.

```sas
data final.DEM (label=&label.);
retain &retain.;
set NEW;
keep &retain.;
run;
```

By placing the retain statement prior to the set statement, SAS will order the columns according to the order in which they were received (RETORD). Similarly, the &retain character string may also be used in the keep statement, assuming the 4 variables were the only ones required for the final dataset. In this example there are only four variables, so the use of the automatically created &retain does not necessarily save time, but when 50 variables or more are necessary, this becomes a very useful utility. Using the STRUCTURE dataset, the label required for the final dataset is automatically obtained.

For most clinical studies, data are stored in several different datasets, often referred to as panels or domains. In the very brief example shown (Figure 1), there are 4 separate datasets needed (DEM, AE, MED, and VIT). Using the same technique as applied to DEM, the process may be generalized to the other three datasets. By replacing the panel specification (DEM) in the previous example with a manually derived macro variable, the process may be generalized to the other datasets.

```sas
%let data=AE;
proc sql noprint;
select n(variable)into :nvar
from permanent.structure where panel="&DATA.";
select variable into :var1-:var%left(&nvar)
from permanent.structure where panel="&DATA.";
select description into :label1-:label%left(&nvar)
from permanent.structure where panel="&DATA.";
select length into :length1-:length%left(&nvar)
from permanent.structure where panel="&DATA.";
select variable into :retain separated by ' '
from permanent.structure where panel="&DATA.";
select ddesc into :describe
from permanent.structure where panel="&DATA.";
quit;
%macro NEWDATA;
data NEW;
%do i=1 %to &nvar.;
  attrib &&var&i.. label"&&label&i.. length=&&length&i..
%end;
set Original.SAS_DATA;
*** Mapping ***;
run;
%mend NEWDATA;
data final.&data. (label=&label.);
retain &retain.;
set NEW;
keep &retain.;
run;
```

A general structure is established that may be applied across various datasets, whether it is 1-2 datasets, or several dozen. As long as specifications are available for every dataset, the process may be used. Admittedly, different clients add various levels of complexity to their desired databases, but given the versatility of SAS for accessing and customizing data, as long as electronic specifications are available, this approach will work.

Not everyone is comfortable with the use of PROC SQL and macro processing. If the programs were applied to a subsequent project managed by a different programmer, interpretation of the programs and debugging could be
unnecessarily problematic. However, with the proper placement of the following option, an “alternative” version of the program without SQL code or macro processing will be created:

```sas
filename mprint "[.location]newprogramname.sas";
options mprint mfile;
```

By placing the FILENAME and OPTIONS statement between the PROC SQL code and the code within the %NEWDATA macro, a copy of the program is created with no SQL code, and all macro variables resolved, which provides a cleaner program, understood by programmers of all skill levels rather quickly:

```sas
data NEW;
  attrib DOB label="BIRTH DATE" length=$20.;
  attrib AGE label="AGE (years)" length=3;
  attrib RACE label="RACE" length=3;
  attrib SEX label="GENDER" length=3;
set Original.SAS_DATA;
*** Mapping ***;
run;
data final.DEM (label=DEMOGRAPHY );
  retain DOB AGE RACE SEX;
set DEM;
  keep DOB AGE RACE SEX;
run;
```

The new program is both easy to read and may be applied to a subsequent project by a programmer of all skill levels. Modifications and de-bugging, if necessary, are apparent and straightforward. Additionally, the time saved can be significant, allowing more attention to be spent on insuring accuracy during the mapping process.

**CONCLUSIONS**

If electronic specifications are available, clinical SAS programmers can exploit these resources to reduce errors and decrease time spent checking structural details. The increased efficiency provided by this approach allows the programmer to accurately map the data from one format to the deliverable format. Very often, when data is submitted, the automatic checks focus on semantics, specifically correct column and sort order, variable names and descriptions, and types or lengths. By exploiting client specifications, revisions of this nature are avoided.

A rather useful option has been presented, that when used appropriately, provides an alternative approach that represents beginner level code with no SQL or macro processing. This representation of the process will be accessible to programmers of any skill level, and provides an easy-to-read and modifiable approach. With the increasing popularity of electronic data submissions to regulatory agencies and the industry’s tendency to strive for flexible standardization, techniques such as these may provide programmers with great timesaving advantage.

**REFERENCES**


Murphy, William C. (2003), *Using a SAS Macro to Document the Database*, Paper 91-28, SUGI 28


AKNOWLEDGMENTS
I would like to thank my colleagues in the Biostatistics and Pharmacokinetics groups in the Department of Clinical Pharmacology at MDS who provided comments on earlier drafts. Their insights and guidance are appreciated.

CONTACT INFORMATION
Your comments and questions are valued and encouraged. Please contact the author at:

Bradford J. Danner
MDS Pharma Services
621 Rose Street
Lincoln, NE 68502
bradford.danner@mdsinc.com
FIGURES

![Image of a diagram depicting the exploitation of available metadata to produce a desired end product and an easy-to-read program to duplicate the process.]

**Figure 1.** A visual representation depicting the exploitation of available metadata to produce a desired end product and an easy-to-read program to duplicate the process.

<table>
<thead>
<tr>
<th>PANEL</th>
<th>VARIABLE</th>
<th>LABEL</th>
<th>RETORD</th>
<th>LENGTH</th>
<th>DDESC</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES</td>
<td>AENUM</td>
<td>ADVERSE EVENT NUMBER</td>
<td>1</td>
<td>$3.</td>
<td>ADVERSE EVENTS : DESCRIPTION</td>
</tr>
<tr>
<td>AES</td>
<td>ADVERS</td>
<td>REPORTED TERM</td>
<td>2</td>
<td>$200.</td>
<td>ADVERSE EVENTS : DESCRIPTION</td>
</tr>
<tr>
<td>AES</td>
<td>LTERM</td>
<td>ADVERSE EVENT LLT</td>
<td>3</td>
<td>$8.</td>
<td>ADVERSE EVENTS : DESCRIPTION</td>
</tr>
<tr>
<td>AES</td>
<td>LLCODE</td>
<td>AE LLT CODE</td>
<td>4</td>
<td>$100.</td>
<td>ADVERSE EVENTS : DESCRIPTION</td>
</tr>
<tr>
<td>AES</td>
<td>PREFTER</td>
<td>PREFERRED TERM</td>
<td>5</td>
<td>$8.</td>
<td>ADVERSE EVENTS : DESCRIPTION</td>
</tr>
<tr>
<td>AES</td>
<td>DHLTCD</td>
<td>ADVERSE EVENT HLT</td>
<td>6</td>
<td>$8.</td>
<td>ADVERSE EVENTS : DESCRIPTION</td>
</tr>
<tr>
<td>AES</td>
<td>DHLTGCOD</td>
<td>ADVERSE EVENT HLGT</td>
<td>7</td>
<td>$8.</td>
<td>ADVERSE EVENTS : DESCRIPTION</td>
</tr>
<tr>
<td>AES</td>
<td>BODYSOC</td>
<td>ADVERSE EVENT PRIMARY SOC</td>
<td>8</td>
<td>$8.</td>
<td>ADVERSE EVENTS : DESCRIPTION</td>
</tr>
<tr>
<td>DEM</td>
<td>DOB</td>
<td>BIRTH DATE</td>
<td>1</td>
<td>$20.</td>
<td>DEMOGRAPHY</td>
</tr>
<tr>
<td>DEM</td>
<td>AGE</td>
<td>AGE (years)</td>
<td>2</td>
<td>3</td>
<td>DEMOGRAPHY</td>
</tr>
<tr>
<td>DEM</td>
<td>RACE</td>
<td>RACE</td>
<td>3</td>
<td>3</td>
<td>DEMOGRAPHY</td>
</tr>
<tr>
<td>DEM</td>
<td>SEX</td>
<td>GENDER</td>
<td>4</td>
<td>3</td>
<td>DEMOGRAPHY</td>
</tr>
<tr>
<td>MED</td>
<td>DAY</td>
<td>DAY</td>
<td>1</td>
<td>$5.</td>
<td>MEDICATION ADMINISTRATION</td>
</tr>
<tr>
<td>MED</td>
<td>SUPPLY</td>
<td>TREATMENT SUB-UNIT</td>
<td>2</td>
<td>$5.</td>
<td>MEDICATION ADMINISTRATION</td>
</tr>
<tr>
<td>MED</td>
<td>MDATE</td>
<td>DRUG DATE AND TIME</td>
<td>3</td>
<td>$20.</td>
<td>MEDICATION ADMINISTRATION</td>
</tr>
<tr>
<td>MED</td>
<td>LOTNO</td>
<td>THERAPEUTIC UNIT NUMBER</td>
<td>4</td>
<td>$6.</td>
<td>MEDICATION ADMINISTRATION</td>
</tr>
<tr>
<td>MED</td>
<td>EXPDA</td>
<td>EXPIRY DATE</td>
<td>5</td>
<td>$20.</td>
<td>MEDICATION ADMINISTRATION</td>
</tr>
<tr>
<td>MED</td>
<td>MEDCOM</td>
<td>COMMENTS</td>
<td>6</td>
<td>$200.</td>
<td>MEDICATION ADMINISTRATION</td>
</tr>
<tr>
<td>VIT</td>
<td>DAY</td>
<td>DAY NUMBER</td>
<td>1</td>
<td>$5.</td>
<td>VITAL SIGNS</td>
</tr>
<tr>
<td>VIT</td>
<td>TIME</td>
<td>SCHEDULE TIME</td>
<td>2</td>
<td>6</td>
<td>VITAL SIGNS</td>
</tr>
<tr>
<td>VIT</td>
<td>VISDAT</td>
<td>EXAMINATION DATE AND TIME</td>
<td>3</td>
<td>$20.</td>
<td>VITAL SIGNS</td>
</tr>
<tr>
<td>VIT</td>
<td>HEIGHT</td>
<td>HEIGHT (cm)</td>
<td>4</td>
<td>3</td>
<td>VITAL SIGNS</td>
</tr>
<tr>
<td>VIT</td>
<td>WEIGHT</td>
<td>WEIGHT (kg)</td>
<td>5</td>
<td>4</td>
<td>VITAL SIGNS</td>
</tr>
<tr>
<td>VIT</td>
<td>BMI</td>
<td>BODY MASS INDEX (kg/m2)</td>
<td>6</td>
<td>3</td>
<td>VITAL SIGNS</td>
</tr>
<tr>
<td>VIT</td>
<td>TEMP</td>
<td>ORAL TEMPERATURE (°C)</td>
<td>7</td>
<td>4</td>
<td>VITAL SIGNS</td>
</tr>
</tbody>
</table>

**Figure 2.** Excerpt from permanent SAS dataset created directly from available specifications.