

Approaches to Data Transfer from Microsoft Excel® to SAS® V9

Hong Qi, Liping Zhang, & Jiannan (Jane) Kang
Merck & Co., Inc., Upper Gwynedd, PA

ABSTRACT

SAS V9 offers some new and flexible ways to transfer data from Excel to SAS. It has improved upon the many limitations in previous SAS versions. This paper summarizes several practical approaches including their undocumented features to data transfer from Excel to SAS V9 and suggests some optimal approaches under different scenarios from the user's point of view. Examples are given with the LIBNAME statement, PROC SQL Pass-Through Facility, IMPORT Procedure, PROC ACCESS, %XLXP2SAS – a SAS V9 macro developed by SAS Institute, and %EXDDE - a macro using the DDE approach developed at Merck & Co., Inc.

INTRODUCTION

The need to transfer data from Excel to SAS is a frequently encountered task. This data process usually brings challenges to SAS users due to the various formats of Excel data such as a column (variable) with mixed character and numeric formats, information across multiple worksheets, the range of data to be transferred, and blank records and extraneous columns (variables) on the sheet.

In SAS V8 and earlier versions, the accuracy of data transfer for Excel data with column(s) in mixed formats is the major issue for most approaches, such as the IMPORT WIZARD/Procedure and PROC SQL ODBC Pass-Through Facility. In SAS V9, this issue has been resolved. In addition, SAS V9 brings some new and flexible features to data transfer, for instance a LIBNAME statement that allows accessing Excel data directly from SAS, the use of the capabilities of the new PC files engine, and more options in the procedure statement (customizing the import dataset).

To understand how to handle the challenges in data transfer from Excel to SAS in SAS V9, the authors experimented with different approaches using various Microsoft Excel data (Microsoft Excel 97-2002 & 5.0/95 Workbook) in Microsoft Office XP, including the LIBNAME statement, PROC SQL Pass-Through Facility, IMPORT Wizard or Procedure, PROC ACCESS, %XLXP2SAS – a SAS V9 macro developed by SAS Institute, and %EXDDE - a macro using the DDE approach developed at Merck & Co., Inc. The following sections provide a description of the input Excel file used in our testing, an overview, the features new and undocumented in SAS V9, the syntax of the code, examples, the best scenarios, cautions, and tips on using each approach.

THE INPUT EXCEL FILE AND THE TEST DESIGN

The input Excel file, shown in Figure 1 through 4 for the four worksheets, is designed to test each data transfer approach in three aspects - accuracy, efficiency and user-friendly features. As shown in the figures, columns (variables) are in various formats - numeric, date, time, character and numeric mixed, character and date mixed. There are some issues that are worth of noting in the figures: the names of columns are in character strings with or without spaces, numbers (12345) and a number plus the same character string (1Score, 2Score and 3Score); the worksheets on the first two figures are identical in terms of format; the variable names on the first three figures are not on the first row of the worksheets; blank columns or rows are hidden in the worksheets in order to display the sparse data, such as Column N to Z and Row 9 to 24 in the worksheet on Figure 3; the worksheet in Figure 3 has a long sheet name with 31 characters and information with LRECL greater than 256. Other test Excel files are omitted here. The accuracy was determined by the correct transfer of data. The efficiency was assessed based on the ability to handle multiple worksheets, extraneous variables and blank records, and the simplicity of the syntax. Features and options of each method were reviewed and considered the feedback in the log file, the volume of the code, flexibility with the range of data, and the conversion of variable names and labels.

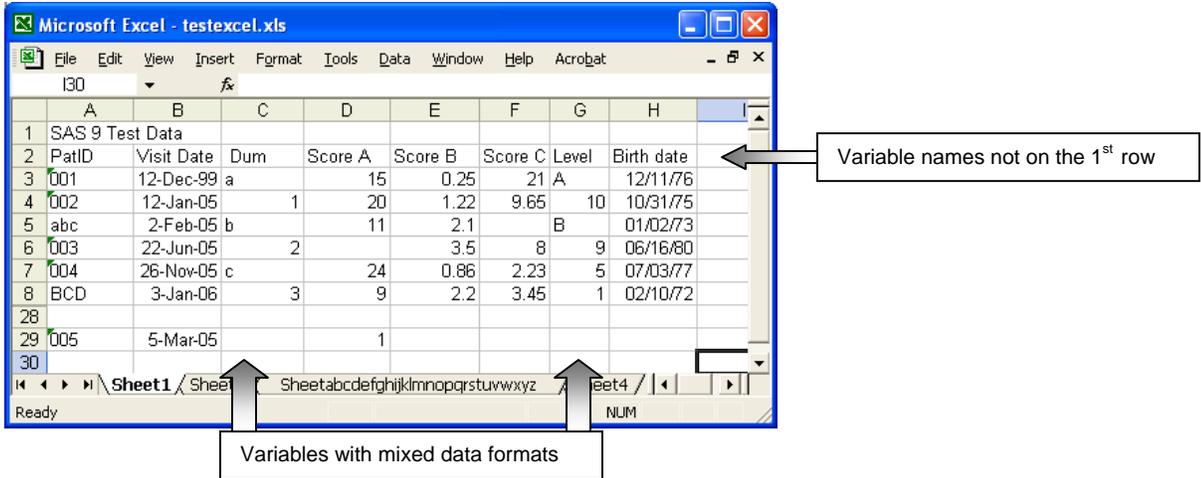


Figure 1. The input Excel data – testexcel.xls (Sheet1)

Worksheets Sheet1 and Sheet2 are identical in format

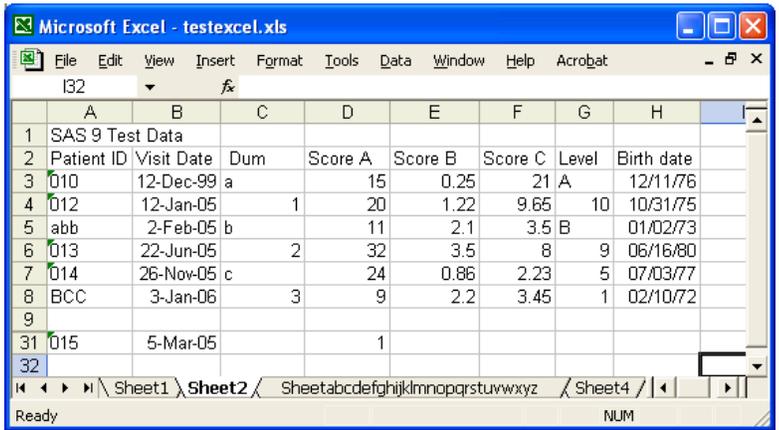


Figure 2. The input Excel data – testexcel.xls (Sheet2)

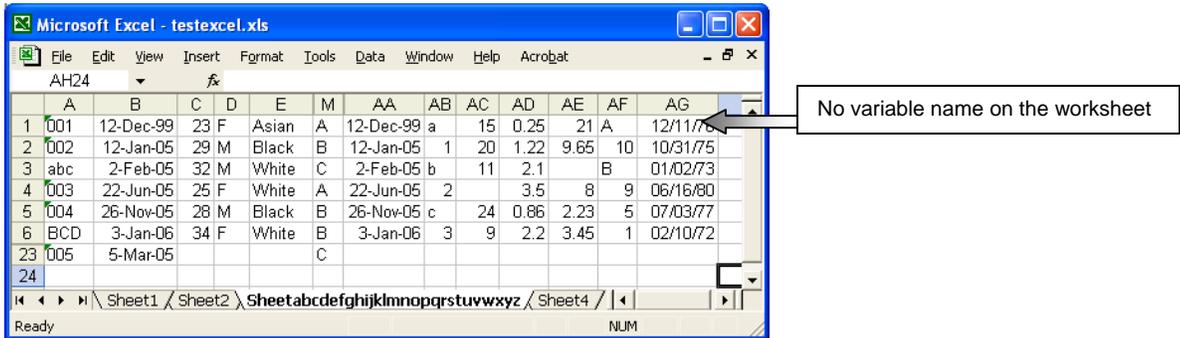


Figure 3. The input Excel data – testexcel.xls (Sheetabcdfghijklmnopqrstuvwxyz)

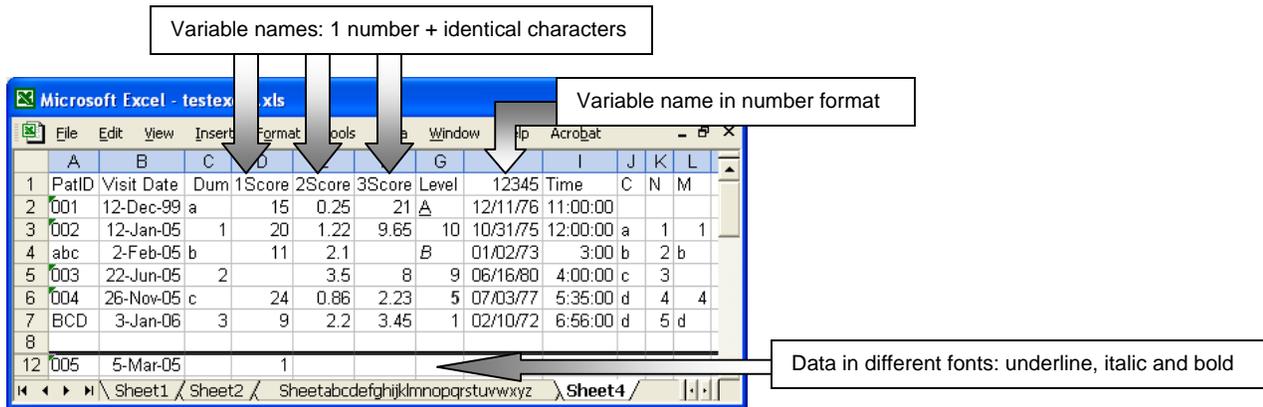


Figure 4. The input Excel data – testexcel.xls (Sheet4)

LIBNAME STATEMENT

As a new feature in SAS V9, the LIBNAME statement provides a direct, transparent, and procedure-free access to data in Excel format. It is available for Microsoft Excel (5, 95, 97, 2000, or 2002) data. The SAS/ACCESS LIBNAME statement extends the SAS global LIBNAME statement to support the association of a libref to Microsoft Excel file that allows referring the spreadsheets directly in a DATA step or SAS procedure as if they were SAS datasets.

The LIBNAME statement permits accurate data transfer for all the data scenarios we tested. With one simple LIBNAME statement, it automatically associates the multiple worksheets to be viewed as multiple corresponding SAS datasets. As a convenient feature, the user does not need to specify the worksheet name or data range of the input Excel file. However, any data manipulation, e.g. removing the dummy data and selecting certain specific range of the data, needs to be handled outside LIBNAME statement. The LIBNAME statement is recommended as a convenient tool for quick data review. When the input Excel data are in standard format, variable names are on the first row and all information needs to be converted, the LIBNAME statement is an ideal approach for bringing data that can be used directly in data step, or SAS procedures.

The following are the LIBNAME statement syntaxes:

1. Assigning SAS data library: **LIBNAME** *libref* <engine-name> <physical-file-name>
2. Clearing (Deassigning) SAS data library: **LIBNAME** *libref* CLEAR | **_ALL_** CLEAR;
3. Writing SAS data library (libraries) attributes to SAS Log: **LIBNAME** *libref* LIST | **_ALL_** LIST;

The LIBNAME statement does not have many options. A few options are commonly included in the LIBNAME statement.

HEADER=YES | **NO** or **GETNAME=YES** | **NO** determines whether the first row of the Excel file is taken as the variable names for the resulting SAS datasets. By default, it is **HEADER=YES** which means that the 1st row of the Excel spreadsheet is taken as the header (variable name) for the dataset. If the data in the 1st row has any blanks in the middle or has leading blanks or numbers, these “illegal characters” are replaced with “_” in the SAS variable names. However, if the 1st row has a missing value or is numeric or date format, the variable names for the import SAS dataset will be: F+ the column number as if **HEADER=NO**.

MIXED=YES | **NO** specifies whether to convert numeric data values into character data values for a column with mixed data types.

SCANTIME=YES | **NO** specifies whether to scan all row values for a DATETIME data type field and automatically determines the TIME data type if only time values (that is, no date or datetime values) exist in the column. **YES** specifies that a column with only time values will be assigned a TIME8. format. **NO** specifies that a column with only time values will be assigned a DATE9. format or DATETIME19 format. This is an undocumented feature for the LIBNAME statement. Alias: **SCAN_TIME=**, and **SCAN_TIMETYPE=**.

The SAS dataset name associated with the Excel spreadsheets under library *libref* will be

Libref. `worksheet name\$`n.

For example, the following code is for assigning the SAS data library *inxls* to the input Excel file:

```
LIBNAME inxls "C:\NESUG2006\testexcel.xls" MIXED=YES HEADER=YES SCANTIME=YES;
```

The SAS datasets that can be referred to in a data step or procedure for the four worksheets are then:

```
inxls.`Sheet1$`n,  
inxls.`Sheet2$`n,  
inxls.`Sheetabcdefghijklmnopqrstuvwxy$`n, and  
inxls.`Sheet4$`n.
```

Figure 5 displays the SAS dataset, *inxls.* `Sheet4\$`n, corresponding to worksheet Sheet4 on the input Excel file. As shown in the dataset, all data information matches that in the original file. The character variable names are also carried over from the Excel file, where the space and numbers are replaced by “_”; column (variable) names 1Score, 2Score, and 3Score in the input file are transferred as *_Score*, *_Score0*, and *_Score1*. The numeric variable name is transferred to F + column number, e.g. F8. The variable labels that are not shown here are exactly the same as the column names on the input file.

	PatID	Visit_Date	_Dum	_Score	_Score0	_Score1	Level	F8	Time	C	N	M
1	001	12DEC1999	a	15	0.25	21	A	11DEC1976	11:00:00	.	.	.
2	002	12JAN2005	1	20	1.22	9.65	10	31OCT1975	12:00:00	a	1	1
3	abc	02FEB2005	b	11	2.1	.	B	02JAN1973	3:00:00	b	2	b
4	003	22JUN2005	2	.	3.5	8	9	16JUN1980	4:00:00	c	3	.
5	004	26NOV2005	c	24	0.86	2.23	5	03JUL1977	5:35:00	d	4	4
6	BCD	03JAN2006	3	9	2.2	3.45	1	10FEB1972	6:56:00	d	5	d
7
8
9
10
11	005	05MAR2005	.	1

Figure 5. SAS dataset *Inxls.Sheet4\$* imported from worksheet Sheet4 in *testexcel.xls* with LIBNAME statement

Some cautions need to be taken when using the LIBNAME statement approach.

1. Before running the LIBNAME statement, the input Excel file must be closed. Otherwise, data in date format will not be imported correctly.
2. A user can assign multiple SAS data libraries to the same Excel file and access the data from the different libraries assigned. However, the Excel file can't be accessed by different users at the same time through the LIBNAME statement. If a user assigns a *libref* to the Excel file when another user has issued a LIBNAME statement to the same Excel file, taking our input file as an example, the following error will be returned:

```
-----  
ERROR: Connect: Unable to IDBInitialize  
ERROR: Error in the LIBNAME statement.  
-----
```

3. SAS datasets in a *libref* associated with an Excel file through a LIBNAME statement have some behavior that differs from that of normal SAS *librefs*. Because these *librefs* refer to database and workbook objects,

such as tables, they are stored in a format that differs from the format of normal SAS data sets. They can't be sorted or updated with a data step. However, the SAS dataset and original Excel file can be changed by PROC DATASETS. The following code, for example, will cause accidental alteration of the Excel file.

```
PROC DATASETS LIBRARY=myxls MEMTYPE=DATA NOLIST;
  DELETE 'Sheet1$'N ;
QUIT ;
```

As a result, all the fields on Sheet1 of the Excel file are deleted.

PROC SQL PASS-THROUGH FACILITY IN SAS V9

The Pass-Through Facility enables interaction with Microsoft Excel (5, 95, 97, 2000, or 2002) data using the data source's SQL syntax without leaving a SAS session. The SQL statements are passed directly to the data source for processing. As a new feature in SAS V9, PROC SQL Pass-Through Facility communicates with Microsoft Excel using the capabilities of the new PC files engine, SAS/ACCESS, instead of going through the predefinition of ODBC Data Source Administrator. Being an alternative to the SAS/ACCESS LIBNAME statement, it provides a direct access to data in Excel format, as well as great scope of control of data imported to SAS.

The Proc SQL Pass-Through Facility permits accurate data transfer for all the data scenarios we tested. Besides transferring data, the PROC SQL Pass-Through Facility also has many flexible features. It is able to subset data by utilizing the SQL "WHERE" clause to limit records and the "SELECT" clause to limit fields. It can apply PROC SQL features in the CREATE TABLE statement in order to best control the attributes of the variables, and create derived variables based on the existing variables from the Excel file. It allows reading multiple identical worksheets in one procedure due to its feature to retrieve sheet names. It is also convenient to use since the Excel file could be open or closed during the transfer process. It is recommended using the PROC SQL Pass-Through Facility as a tool for data import and process when customized SAS dataset is needed.

The following is the basic syntax for PROC SQL Pass-Through Facility:

```
PROC SQL;
  CONNECT TO EXCEL (PATH="path-name\excel-file-name.xls");
  CREATE TABLE SAS-data-set-name AS
  SELECT * FROM CONNECTION TO EXCEL
    (SELECT * FROM [Sheet-name$]);
  DISCONNECT FROM EXCEL;
QUIT;
```

Since it is an undocumented feature, it is worth mentioning here that the "*Sheet-name*" in the "[]" must be followed by a "\$".

PROC SQL has many options. Three of its commonly used options, **HEADER=YES|NO** or **GETNAME=YES|NO** and **Mixed=YES|NO**, are the same as those of the LIBNAME statement. Below are the descriptions of other commonly used options.

FEEDBACK|NOFEEDBACK specifies whether to write a statement to the SAS log that expands the query.

USE_DATATYPE=YES|NO specifies whether to use DATE. format for datetime columns in the data source table while importing data from Microsoft Excel workbook. By default, it is **USE_DATATYPE=YES** for Microsoft Excel workbook, which means that the SAS DATE format is assigned for datetime columns in the data source table. **USE_DATATYPE=NO** specifies that the SAS DATETIME format is assigned for datetime columns in the data source table. Alias: **USE_DATE=** and **USEDATE=**.

ERRORSTOP|NOERRORSTOP specifies whether PROC SQL should stop executing after an error.

The following code transfers Sheet4 from the input Excel file into a SAS dataset, *Sql_sh4*, shown in Figure 6. Note that it only contains records with non-missing *PatID*. All its variable names are identical to those in the

dataset generated with LIBNAME statement (Figure 5). Variables with prefix "f_" are the derived variables with the same date or time format as those in the original Excel worksheet.

```

PROC SQL;
CONNECT TO EXCEL (PATH="C:\NESUG2006\testexcel.xls" MIXED=YES
                 USE_DATATYPE=NO);
CREATE TABLE sql_sh4 as
SELECT * ,
      DATEPART(Visit_Date) format=date9.
                          label="Visit Date"
                          as f_visit_date,
      DATEPART(F8)        format=date9.
                          label="12345"
                          as f_F8,
      TIMEPART(Time)      format=time9.
                          label="Time"
                          as f_time
FROM CONNECTION TO EXCEL
      (SELECT * FROM [Sheet4$]
      WHERE PatID IS NOT NULL);
DISCONNECT FROM EXCEL;
QUIT;

```

Some cautions need to be taken when using the PROC SQL Pass-Through Facility approach.

1. When reading multiple worksheets, this approach is not able to work around or skip the first description row.
2. When the format of time value in the Excel file shows as "hh:mm:ss", in order to read in all the time or date value correctly from Excel file, the user needs to specify each time or date field, using either the DATEPART or TIMEPART in the Select statement. And then, in the SAS code, use the Connection option "USE_DATATYPE=NO" so that all the fields will be brought in as time or date values.
3. By default, the character variable always has format of \$255.
4. The syntax can be very tedious when this approach is used for generating a customized dataset.

	PatID	Visit_Date	Dum	_Score	_Score0	_Score1	Level	F8	Time	C	N	M	f_visit_date	f_F8	f_time
1	001	12DEC1999:00:00:00	a	15	0.25	21	A	11DEC1976:00:00:00	30DEC1899:11:00:00	.	.	.	12DEC1999	11DEC1976	11:00:00
2	002	12JAN2005:00:00:00	1	20	1.22	9.65	10	31OCT1975:00:00:00	30DEC1899:12:00:00	a	1	1	12JAN2005	31OCT1975	12:00:00
3	abc	02FEB2005:00:00:00	b	11	2.1	.	B	02JAN1973:00:00:00	30DEC1899:03:00:00	b	2	b	02FEB2005	02JAN1973	3:00:00
4	003	22JUN2005:00:00:00	2	.	3.5	8	9	16JUN1980:00:00:00	30DEC1899:04:00:00	c	3	.	22JUN2005	16JUN1980	4:00:00
5	004	26NOV2005:00:00:00	c	24	0.86	2.23	5	03JUL1977:00:00:00	30DEC1899:05:35:00	d	4	4	26NOV2005	03JUL1977	5:35:00
6	BCD	03JAN2006:00:00:00	3	9	2.2	3.45	1	10FEB1972:00:00:00	30DEC1899:06:56:00	d	5	d	03JAN2006	10FEB1972	6:56:00
7	005	05MAR2005:00:00:00	.	1	05MAR2005	.	.

Figure 6. SAS dataset Sql_sh4\$ imported from worksheet Sheet4 in testexcel.xls with PROC SQL Pass-Through Facility

IMPORT PROCEDURE

The IMPORT Procedure reads data from an external data source (e.g. Excel files) and writes it to a SAS data set. Excel files can be imported in two ways: (1) use PROC IMPORT statement or (2) use the Import Wizard, which is

a windowing tool that guides you through the steps. It is available for Microsoft Excel (4, 5, 95, 97, 2000, or 2002) data. Compared to the early versions, the IMPORT Procedure in SAS V9 has more features/options. For example, in Data Source Statement, options such as MIXED, SCANTEXT, TEXTSIZE, SCANTIME, and USEDATE are available to make the IMPORT Procedure more powerful and flexible. The following discussion focuses on the PROC IMPORT statement.

The IMPORT Procedure permits accurate data transfer for all the data scenarios we tested. As a major advance in SAS V9, the IMPORT Procedure has the ability to handle Excel columns in mixed formats. Like Proc SQL, it allows the Excel file to be open or closed during the transfer process. The IMPORT procedure can handle most of the Excel files; and the syntax is relatively easy. The IMPORT Procedure is recommended as a tool for quick data review, importing and processing for data from a single worksheet in the Excel file.

The PROC IMPORT syntax is:

```
PROC IMPORT;
  DATAFILE="filename" | TABLE="tablename"
  OUT=<libref.>SAS-data-set <(SAS-data-set-options)>
    <DBMS=identifier><REPLACE>;
  <data-source-statement(s)>;
RUN;
```

PROC IMPORT has quite a few options. Four of its commonly used options, **HEADER=YES|NO** or **GETNAME=YES|NO**, **MIXED=YES|NO**, and **SCANTIME=YES|NO** are the same as those of the LIBNAME statement. The following lists a few more commonly used:

SAS-data-set-options specify data set options. For example, to assign a password to the resulting SAS data set, you can use the ALTER=, PW=, READ=, or WRITE= data set options, or import only data that meets a specified conditions. (e. g. you can use WHERE= data set option)

DBMS=identifier specifies the type of data to import, e.g. DBMS=EXCEL or DBMS=XLS

TEXTSIZE=1 to 32767 specifies the field length that is allowed for importing MS Excel 97, 2000 or 2002 memo fields. The default is 1024.

RANGE="range-name" | "absolute-range" specifies the range to be included

Range-name - a name assigned to represent a range

Absolute-range - identifies the top left cell that begins the range and the bottom right cell that ends the range, e.g. C9:F12. For Excel 97 and above, you can include the worksheet name with an absolute range, such as range="North B\$C9:F12"

SCANTEXT=YES|NO scans the length of the text data for a data source column and uses the length of the longest string data found as the SAS column width. However, if the length found is greater than what is specified in the TEXTSIZE= option, then the smaller value specified in TEXTSIZE= will be applied as the SAS variable width.

SHEET=spreadsheet-name identifies a particular worksheet in a group of worksheets.

USEDATE=YES|NO uses DATEw. format for date/time columns when YES; uses DATETIME. format when NO.

The following example illustrates how to use the IMPORT Procedure to import *testexcel.xls* (sheet4):

```
PROC IMPORT OUT=Impt_sh4
  DATAFILE="C:\NESUG2006\testexcel.xls"
  DBMS=EXCEL REPLACE;
  RANGE="Sheet4$A1:L28";
  GETNAMES=YES;
  MIXED=YES;
  SCANTEXT=YES;
  USEDATE=YES;
  SCANTIME=YES;
```

```
RUN;
```

This generates the SAS dataset `Impt_sh4` which is identical to `Sheet4` generated with `LIBNAME` statement.

Some Cautions need to be taken when using the `IMPORT` Procedure:

1. It can only import one worksheet at a time, but not multiple in one `PROC IMPORT` procedure.
2. If `MIXED=YES` is not specified for columns in mixed numeric and character data type, `PROC IMPORT` scans the column and uses the most popular one on the 1st 8 rows as the data type.
3. If the `REPLACE` option is not specified, `PROC IMPORT` will not overwrite an existing data set.

ACCESS PROCEDURE IN SAS V9

The `ACCESS` Procedure creates an access descriptor to store data from an Excel file and writes it to a SAS data set. By default, it uses Microsoft Excel 5 data, but it is available for Microsoft Excel (4, 5, 95, 97, 2000, or 2002) data. As in the early versions, the `ACCESS` Procedure in SAS V9 is still a powerful and flexible approach for data transfer from Excel to SAS.

The `ACCESS` Procedure permits accurate data transfer for all the data scenarios we tested. As convenient features, it allows defining variable attributes, subsetting data, dropping records and skipping rows within the procedure. Multiple worksheets can be imported by one procedure with multiple `CREATE` statements. It is recommended using the `ACCESS` Procedure when some data manipulation needs to be done.

`PROC ACCESS` syntax is:

```
LIBNAME libref "C:\NESUG2006";

PROC ACCESS DBMS=XLS;
  CREATE libref.descriptor-name.ACCESS;          /* Create ACCESS descriptor */
  PATH="C:\NESUG2006\testexcel.xls";
  CREATE libref.SAS-data-name.VIEW;             /* Create dataset          */
  SELECT ALL;
RUN;
```

`PROC ACCESS` has many options. Some of its commonly used options, `GETNAME=YES|NO`, `Mixed=YES|NO`, `SCANTIME=YES|NO`, `Range=`, and `path=` are the same as those of the `LIBNAME` statement and `IMPORT` Procedure. The following lists a few other more commonly used ones:

`CREATE libref.member-name.ACCESS|VIEW` creates a SAS/ACCESS descriptor file.

`SCANTYPE=YES|NO|Y|N|<number-of-row>` finds the most common data type and format in each column in a specified number of rows in an XLS worksheet in order to generate the default SAS format.

`SKIPROWS=<number-of-row-to-skip>` specifies the number rows to ignore when reading data from the Excel file.

`WORKSHEET=Worksheet-name` identifies the worksheet to read, `Sheet1` by default.

`FORMAT column-identifier=SAS-format-name` changes a SAS format for a PC file column.

`LIST ALL|VIEW|column-identifier` lists columns in the descriptor and gives their information.

`SUBSET selection criteria` add or modify selection criteria for a view descriptor.

The following is an example of how to use `Proc Access` to import `testexcel.xls` (sheet4):

```
LIBNAME padir "C:\NESUG2006";
PROC ACCESS DBMS=XLS;
```

```

CREATE padir.sh4_desc.ACCESS;
  PATH="C:\NESUG2006\testexcel.xls";
WORKSHEET=sheet4;
  GETNAMES=YES;
  SCANTYPE=YES;
  MIXED=YES;
  FORMAT 2=mmddyy10.;
SUBSET WHERE patid is not null
  or visitdat is not null
  or dum is not null
  or z1score is not null
  or z2score is not null
  or z3score is not null
  or level is not null
  or var7 is not null;
LIST ALL;

CREATE padir.sh4_data.view;
SELECT ALL;
RUN;

```

This generates the SAS dataset, Padir.Sh4_data, as shown in Figure 7. As a result of SUBSET, the blank records are removed. Different from other approaches bringing the column names to SAS, the column names with "number + identical characters" (1score, 2score, and 3score) are replaced by "z + the original column name", the column name with space is replaced by concatenated first eight characters, and the column name (12345) in number format is replaced by "var+ the number of the column (var7).

Some Cautions need to be taken when using the ACCESS Procedure:

1. SAS dataset is generated in two steps - generating the SAS/ACCESS descriptor and view.
2. The Excel file must be closed when the PROC ACCESS is executing.
3. Sheet1 is imported by default if WORKSHEET is not specified.

	PATI	VISITDAT	DUM	Z1SCORE	Z2SCORE	Z3SCORE	LEVEL	VAR7	TIME	C	N	M
1	001	12/12/1999	a	15	0.25	21	.	12/11/76	11:00:00	.	.	.
2	002	01/12/2005	1	20	1.22	9.65	10	10/31/75	12:00:00	a	1	1
3	abc	02/02/2005	b	11	2.1	.	.	01/02/73	3:00:00	b	2	b
4	003	06/22/2005	2	.	3.5	8	9	06/16/80	4:00:00	c	3	.
5	004	11/26/2005	c	24	0.86	2.23	5	07/03/77	5:35:00	d	4	4
6	BCD	01/03/2006	3	9	2.2	3.45	1	02/10/72	6:56:00	d	5	d
27	005	03/05/2005	.	1

Figure 7. SAS dataset Padir.Sh4_data imported from worksheet Sheet4 on testexcel.xls with PROC Access

SAS V9 MACRO %XLXP2SAS

%Xlxp2sas is a SAS V9 macro designed, together with SAS XMLMap, specifically for *Excel XML* data by SAS Institute. It provides an automatic way to import multiple Excel worksheets with SAS supported by the SAS XML LIBNAME Engine (SXLE) and the SAS XMLMap. This operation is available for Microsoft Excel 2002 or later versions since the Excel file must be saved as an XML file before converting into SAS data with the macro, and only Excel 2002 and later versions support saving Excel spreadsheets in XML format. The macro has six

parameters and only two of them are required for running the macro. The macro and SAS XMLMap can be downloaded from support.sas.com/saspresents.

%Xlpx2sas permits accurate data transfer for the following data scenarios we tested: data with columns (variables) in the formats of character, numeric, mixed character and numeric, and mixed character and date; data in worksheet with long sheet name; and data with record length greater than 256. With one simple macro call, it automatically imports the multiple worksheets to multiple corresponding SAS datasets. As a convenient feature like the LIBNAME statement, the user does not need to specify the worksheet names or data range of the input Excel file in order to transfer all the information on all the worksheets. It also has built-in logic checks to inform the user what is missing in the macro call. %Xlpx2sas is recommended as a tool of data transportation for data in multiple worksheets and in standard format with features such as no any missing column (variable) names if populated on the 1st row, columns (variables) without date- or time-only format, and all information needs to be converted.

The following is the %xlpx2sas syntax:

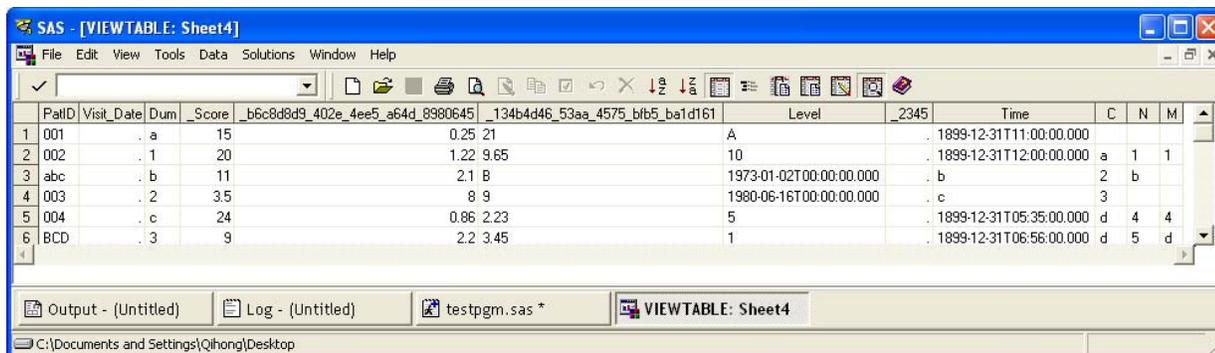
```
%xlpx2sas(excelfile=, /* REQUIRED. Name and path for the input Excel XML file          */
          mapfile  =, /* REQUIRED. Name and path of the SAS XMLMap                    */
          library =, /* Name of the SAS library for the imported tables, WORK by default    */
          haslabels=, /* Have column labels in the first row, Y by default */
          cleanup =, /* Delete the temporary SAS files and de-assign FILEREFs, Y by default */
          verbose = /* Control the debugging information written to the SAS Log, N by default */);
```

Below are the few steps that need to be followed to import the Excel spreadsheets into SAS tables with %xlpx2sas.

1. Save the Excel file as a XML spreadsheet (*.xml).
2. Download the XMLMap (ExcelXP.map) and macro (in LoadXL.sas) and save them to the platform where SAS is installed.
3. Submit the following code for example,

```
%INC 'C:\NESUG2006\loadxl.sas';
%xlpx2sas(excelfile=C:\NESUG2006\testexcel.xml,
          mapfile  =C:\NESUG2006\excelxp.map);
```

The imported SAS datasets are then placed in the WORK library as **sheet1**, **sheet2**, **sheetabcdefghijklmnopqrstuvwxyz**, and **sheet4**.



PatID	Visit_Date	Dum	Score	_b6c8d8d9_402e_4ee5_a64d_8980645_134b4d46_53aa_4575_bfb5_ba1d161	Level	_2345	Time	C	N	M
1	001	. a	15	0.25 21	A	.	1899-12-31T11:00:00.000			
2	002	. 1	20	1.22 9.65	10	.	1899-12-31T12:00:00.000	a	1	1
3	abc	. b	11	2.1 B	1973-01-02T00:00:00.000	.	b	2	b	
4	003	. 2	3.5	8 9	1980-06-16T00:00:00.000	.	c	3		
5	004	. c	24	0.86 2.23	5	.	1899-12-31T05:35:00.000	d	4	4
6	BCD	. 3	9	2.2 3.45	1	.	1899-12-31T06:56:00.000	d	5	d

Figure 8. SAS dataset Sheet4 imported from worksheet Sheet4 in testexcel.xls with %xlpx2sas

Figure 8 displays the SAS dataset, Sheet4, corresponding to worksheet Sheet4 in the input Excel file. As shown in the dataset, the 1st and 3rd variables (PatID and Dum) match the corresponding columns in the original file that contain character and mixed formats. But the Excel columns in date or time only format are not handled properly, i.e. the 2nd and 8th variables (Visit_Date and _2345), from the columns containing date-only format (Visit Date and 12345), have all information missing; and the 9th variable (Time), from the column in time-only format, does not get the right information. Meanwhile, some variables, the 4th, 5th, 6th, 7th, etc., have incorrect information that seems due to the blank cells in the Excel data. The non-identical character column names, such as PatID, Visit Date, Dum, etc. from Excel are carried over to SAS dataset, where the space and numbers are replaced by “_”. The identical character column names, 1Score, 2Score, and 3Score, in the input file are transferred as _Score, _b6c8d8d9_402e_4ee5_a64d_8980645, and _134b4d46_53aa_4575_bfb5_bald161. The variable labels that are not shown here are exactly the same as the column names on the input file.

	COLUMN001	COLUMN002	COLUMN003	COLUMN004	COLUMN005	COLUMN006	COLUMN007	COLUMN008	COLUMN009	COLUMN010
1	PatID	Visit Date	Dum	1Score	2Score	3Score	Level	12345	Time	
2	001	1999-12-12T00:00:00.000	a	15	0.25	21	A	.	1899-12-31T11:00:00.000	
3	002	2005-01-12T00:00:00.000	1	20	1.22	9.65	10	.	1899-12-31T12:00:00.000	a
4	abc	2005-02-02T00:00:00.000	b	11	2.1	B	1973-01-02T00:00:00.000	.	b	2
5	003	2005-06-22T00:00:00.000	2	3.5	8	9	1980-06-16T00:00:00.000	.	c	3
6	004	2005-11-26T00:00:00.000	c	24	0.86	2.23	5	.	1899-12-31T05:35:00.000	d
7	BCD	2006-01-03T00:00:00.000	3	9	2.2	3.45	1	.	1899-12-31T06:56:00.000	d

Figure 9. SAS dataset Sheet4 imported from worksheet Sheet4 on testexcel.xls with %xlxp2sas

Figure 9 shows the SAS dataset, Sheet4, imported from the same input Excel file when HASLABELS=N is specified in the macro call. In this case, the 1st row on the Excel file is considered as data values rather than variable names, dates in the 2nd column are converted to “date + T00:00:00”, but the date in the 8th column are missing. “COLUMN001”, “COLUMN002” and so on are the variable names for the SAS dataset, and “Column 001”, “Column 002” and so on are the variable labels.

Some cautions need to be taken when using the %xlxp2sas approach.

1. The data in the input Excel spreadsheet must be fairly rectangular to prevent unpredicted results
2. It is not suggested to use for the import of Excel Spreadsheets with columns in the date-only or time-only format as discussed previously.
3. %xlxp2sas has limitation in handling some (not all) missing column names on the first row which might not be a usual situation. Figure 10 gives a spreadsheet with missing column name on the 4th column. And Figure 11 shows the resulting SAS dataset generated with %xlxp2sas that replaces the original missing name for the 4th column with that for the 5th column, and loss of information from the 5th column.

	A	B	C	D	E	F	G	H
1	t1	t2	t3		t5			
2		12		26	29	20		
3		23		27	30	20		
4	24-'#@	25		28	31	20		
5								

Figure 10. The worksheet MissText on txlxp2sas.xml

	t1	t2	t3	t4	t5
1	12	.	26	29	
2	23	.	27	30	
3	24~&##%#@	25	28	31	

Figure 11. SAS dataset imported from worksheet MissText in txlpx2sas.xml with %xlpx2sas

MACRO %EXDDE

%Exdde is a macro employing the DDE approach developed in SAS V8 at Merck & Co. Inc. The design of this macro is guided by the principles of accuracy, efficiency and convenience. It provides an automatic way to import multiple Excel worksheets in various formats. This operation is available for Microsoft Excel (5-2002) data. The macro contains seventeen parameters, three of them are required, and the rest are optional and provide further control for the input Excel file and the output SAS datasets. All its features remain the same in SAS V9.

%Exdde permits accurate data transfer for all the data scenarios we tested: data with columns in different formats, including character, numeric, date, time, mixed character and numeric, and mixed character and date; the sparse data; data in worksheet with long sheet name; and data with record length greater than 256. With a simple macro call, it automatically imports and combines multiple identical worksheets into one corresponding SAS dataset. The macro has many efficient and convenient features, including few required parameters, flexibility for the user to specify the variable names and formats in a one-to-one correspondence, automatically removing missing records, options to delete dummy records, and flagging the worksheet origin before generating the output SAS dataset. It also has built-in logic checks to inform the user about various problems in the macro call. The authors recommend using %exdde as a tool of data transportation for any data format, especially for multiple identical worksheets that need to be combined into one SAS dataset, for data with selected range of information that need to be transferred, and when some data manipulation such as removing extraneous variables and records is needed.

Following is the syntax of %exdde:

```
%EXDDE(/** 1. Parameters related to the input Excel file ***/
    Excel = , /* = ON, the Excel file is open; otherwise, the macro opens the Excel file */
    Xlsex = , /* Location of Microsoft Excel on the C: drive */
    Rawdir = , /* REQUIRED. Input Excel file directory */
    File = , /* REQUIRED. Input Excel file name */
    Sheet = , /* Worksheet name(s), e.g., Sheet1|Sheet2. Sheet1 by default. */
    Lrecl = , /* Length of the Excel file, default is 256 */
    StartR = , /* Row number of the 1st cell to read */
    StartC = , /* Column number of the 1st cell to read */
    EndR = , /* Row number of the last cell to read */
    EndC = , /* Column number of the last cell to read */

    /** 2. Parameters related to the output SAS dataset ***/
    Var = , /* Variable list */
    Fmt = , /* Format of the variables */
    Dum = , /* Extraneous variables need to be dropped */
```

```

Flag    =,    /* Flag for the sub-sheet origin of the data                */
Outdata=,    /* REQUIRED. Output SAS data set name                            */

/**** 3. Process related parameters ****/
Sleep  =,    /* Seconds that SAS is suspended from execution while opening up the Excel file */
Debug  =     /* Whether to keep intermediate data sets, default=NO                          */);

```

Note that only three parameters, **Rawdir**, **File** and **Outdata**, must be specified in order to run the macro. For example, the following code generates SAS dataset, Sheet1 from the worksheet Sheet1 in the input Excel file, shown in Figure 12. In this dataset, the blank records between Row 8 and 29 in the original Excel worksheet have been removed by the macro.

```

%INC "C:\NESUG2006\exdde.sas";
%exdde(rawdir =%str(C:\NESUG2006),
       file   =testexcel,
       outdata=Sheet1);

```

	col1	col2	col3	col4	col5	col6	col7	col8	col9
1	SAS 9 Test Data								
2	PatID	Visit Date	Dum	Score A	Score B	Score C	Level	Birth date	
3	001	12-Dec-99	a	15	0.25	21	A	12/11/76	
4	002	12-Jan-05	1	20	1.22	9.65	10	10/31/75	
5	abc	2-Feb-05	b	11	2.1		B	01/02/73	
6	003	22-Jun-05	2		3.5	8	9	06/16/80	
7	004	26-Nov-05	c	24	0.86	2.23	5	07/03/77	
8	BCD	3-Jan-06	3	9	2.2	3.45	1	02/10/72	
9	005	5-Mar-05	1						

FIGURE 12. SAS Dataset Imported from Worksheet Sheet1 in testexcel.xls with %exdde

To combine multiple identical worksheets, e.g. Sheet1 and Sheet2 in Testexcel.xls, into one SAS dataset with customized features, more parameters need to be specified, for instance,

```

%exdde(Rawdir =%str(C:\NESUG2006),
       File   =testexcel,
       Sheet  =sheet1|sheet2,
       StartR =3,
       EndR   =31,
       Var    =PatID Visit_Date Dum ScoreA ScoreB ScoreC Level Birth_Date,
       Fmt    =$3. date9. $1. 3. 5. 5. $2. $8.,
       Flag   =001|002,
       Dum    =Dum Level,
       outdata=Sheet1_2);

```

This macro call creates a combined SAS dataset demonstrated in Figure 13. The dataset is the combination of Sheet1 and Sheet2 from the input Excel file. All data values are imported as being populated on the Excel file along with the variable attributes specified in the macro; it has a variable, FLAG, identifying the origin of the worksheet; and it only includes necessary variables and non-blank records.

Some cautions need to be taken when using %exdde:

1. If no worksheet is specified in the macro, the macro only imports `Sheet1` by default. Multiple worksheet names need to be specified only when they are identical and are to be combined into one SAS dataset. '|' should be used to separate each worksheet name.
2. By default, the macro automatically imports the first 101 rows, and 30 columns in the Excel file into SAS, with the default variable names `col1-col30` and format `$30.` for all the variables.
3. Variable labels or value labels (format) are not supported.

	PatID	Visit_Date	ScoreA	ScoreB	ScoreC	Birth_Date	flag
1	001	12DEC1999	15	0	21	12/11/76	001
2	002	12JAN2005	20	1	10	10/31/75	001
3	abc	02FEB2005	11	2	.	01/02/73	001
4	003	22JUN2005	.	4	8	06/16/80	001
5	004	26NOV2005	24	1	2	07/03/77	001
6	BCD	03JAN2006	9	2	3	02/10/72	001
7	005	05MAR2005	1	.	.	.	001
8	010	12DEC1999	15	0	21	12/11/76	002
9	012	12JAN2005	20	1	10	10/31/75	002
10	abb	02FEB2005	11	2	4	01/02/73	002
11	013	22JUN2005	32	4	8	06/16/80	002
12	014	26NOV2005	24	1	2	07/03/77	002
13	BCC	03JAN2006	9	2	3	02/10/72	002
14	015	05MAR2005	1	.	.	.	002

FIGURE 13. SAS Dataset Imported from Worksheet Sheet1 and Sheet2 in testexcel.xls with %exdde

CONCLUSIONS

SAS V9 brings many features to transfer data from Microsoft Excel to SAS. Each approach has its unique features, as well as some limitations. Therefore, cautions need to be taken in the selection of the approach to import Excel files to SAS. Among all the approaches, the LIBNAME statement provides the easiest syntax to use, but it does not have any data manipulation features. The LIBNAME statement, Proc SQL Pass-Through, IMPORT and ACCESS Procedures, and %exdde are the approaches that can handle all data types in the input Excel file. The LIBNAME statement and %xlxp2sas are similar in transferring multiple Excel worksheets into multiple corresponding SAS datasets. The Proc SQL Pass-Through Facility and %exdde have the flexibility to combine multiple worksheets into one dataset. Figure 14 gives a brief summary and comparison of each approach (Figure 14 to be completed). Building a user-friendly macro with approaches provided in SAS can make the data transfer from Excel to SAS easy.

REFERENCES:

Base SAS V9® 9.1.3 SAS Help and Documentation.

DelGobbo, Vincent (2006) "Creating AND Importing Multiple-Sheet Excel Workbooks the Easy Way with SAS®" in Proceedings of the Thirty-One Annual SAS Users Group International Conference". Paper115-31.

Qi, Hong (2004) "A Practical Approach to Transferring Data from Microsoft Excel® to SAS® in Pharmaceutical Research". Paper 028-29

ACKNOWLEDGMENTS

We thank our manager Allan Glaser for his support and encouragement and Dr. Gregory Golm for reviewing the paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author by mail or e-mail at the following address:

Hong Qi
Merck & Co. Inc.
351 North Sumneytown Pike, UG1D-38
PO Box 1000
North Wales, PA 19454-1099

Work Phone: 267-305-7589
Fax: 267-305-6474
Email: hong_qi@merck.com

Liping Zhang
Merck & Co. Inc.
351 North Sumneytown Pike, UG1D-38
PO Box 1000
North Wales, PA 19454-1099

Work Phone: 267-305-7980
Fax: 267-305-6474
Email: liping_zhang@merck.com

Jannian (Jane) Kang
Merck & Co. Inc.
351 North Sumneytown Pike, UG1D-38
PO Box 1000
North Wales, PA 19454-1099

Work Phone: 267-305-7426
Fax: 267-305-6474
Email: hong_qi@merck.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 trademarks of their respective companies.