The Little Engine That Could: Using EXCEL LIBNAME Engine Options to Enhance Data Transfers between SAS® and Microsoft® Excel Files

William E Benjamin Jr, Owl Computer Consultancy, LLC

Abstract

Many people are not aware that the SAS ® Access for PC Files product will allow SAS Programmers to access an Excel spreadsheet in much the same way as any other SAS file. There are of course some restrictions, but there are also a lot of options that help remove some of the bumps in the road. The LIBNAME statement allows the user to define an Excel file in SAS terms and gives the programmer access to LIBNAME and data set options to control how the Excel file is defined, accessed, and yes even how the data will be formatted. This paper will describe some of those options.

Introduction

Libnames, they are the most common way for SAS programmers to access data. Most programmers only use libnames to access SAS data files. When external data is required the data is converted to a text file and input through a Filename connection of some sort. However, SAS has a product called SAS ® / Access for PC File Formats that will let users read and write data files for Microsoft ® Windows applications. The most popular of these applications and the focus of this paper is Microsoft ® Excel. SAS is a language of defaults; the runtime system looks to see how a variable is first used then assigns a default type to the variable. The programmer types “PROC PRINT; RUN;” and the last created SAS Dataset is sent to the default printer. The SAS language also gives the programmer choices about the code The ATTRIB command assigns either a numeric or a character type, and changes the size of the variable to a small numeric with a limited range of values, or a large character variable bigger than the default. LIBNAME commands are no different; they have options to change the behavior of the input/output processes. This paper addresses some options available for reading and writing data to and from EXCEL files using SAS.

Let Us Look at the Defaults First

Part of the syntax (SAS Version 9.2) for accessing Excel files is as follows:

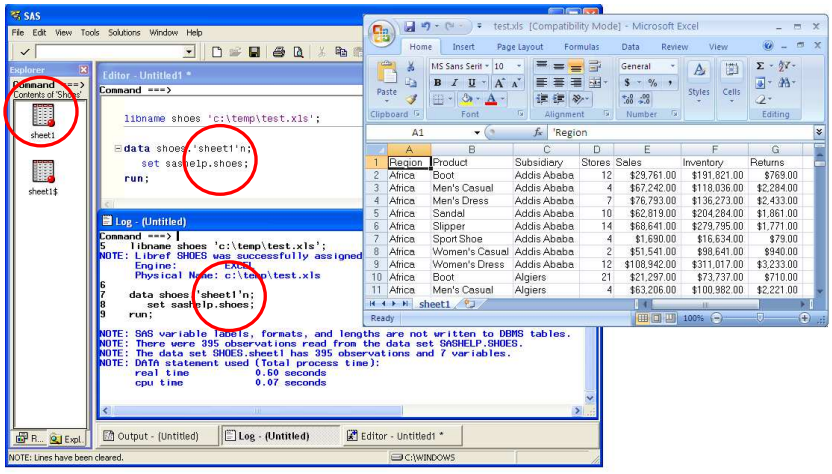
```
LIBNAME libref <engine> <physical-file-name>;
```

That looks pretty much like every other LIBNAME definition. So let’s make it a little more Excel friendly.

```
LIBNAME xls_data EXCEL 'C:\WUSS_2009\My_Excel_File.xls';
```

Now how about putting some data into the file, this will create a file that is made easily and everyone can have the same file. A SAS Code syntax construct called a name literal is used in the following code. (a name literal is quoted text followed by an “n” as in the following ‘Name_Literal’n)

```
Data xls_data.'sheet1'n;
   Set sashelp.shoes;
Run;
```



Because the space in this paper is limited only one or two examples of each type of option will be shown, the others will be left for you to explore. The LIBNAME gets its power from the fact that it gives the programmer access to the defaults, AND ALLOWS THEM TO BE CHANGED. Now let us re-examine the syntax for the LIBNAME command as it applies to Microsoft Excel files. (Other options are also available for other file types.)

LIBNAME `libref <engine> <physical-file-name>`
`<SAS/ACCESS-engine-connection-options>`
`<SAS/ACCESS-libname-options>;`

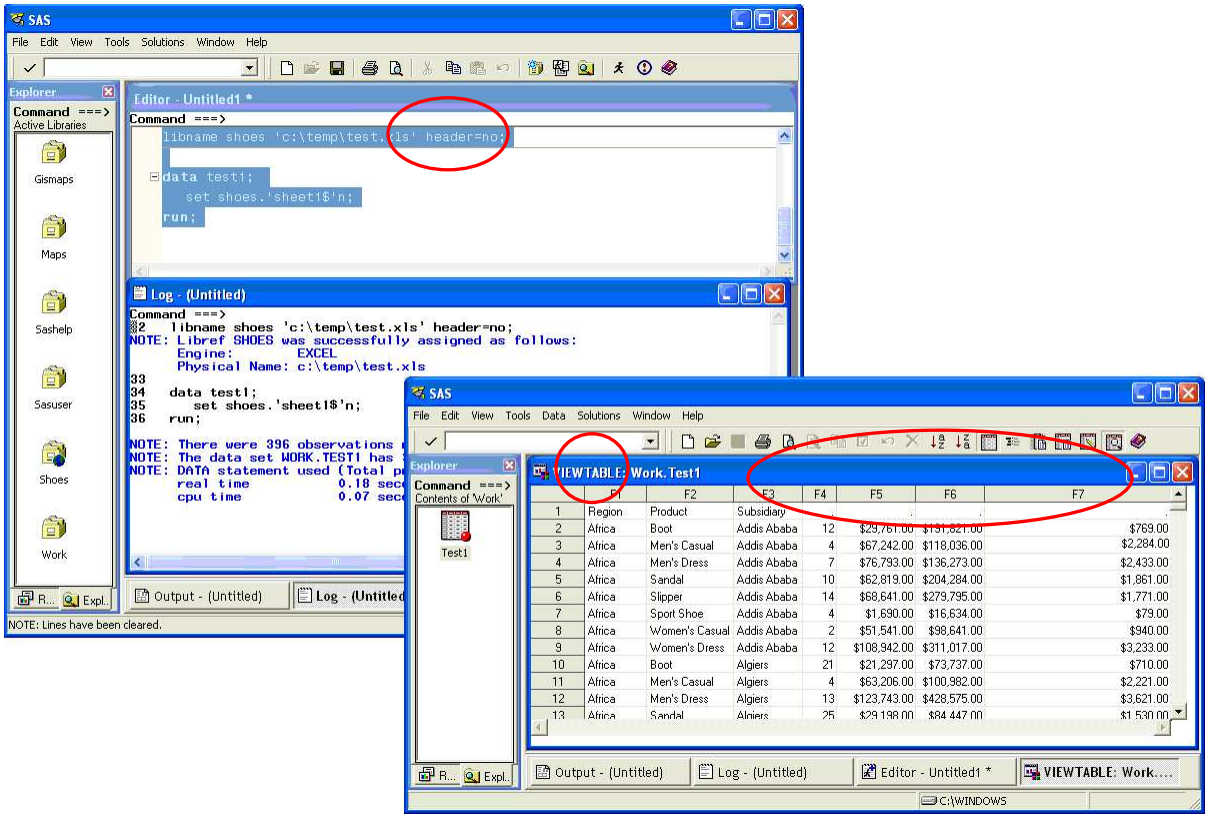
Now we see ”engine-connection-options” and “libname-options”, these are both little used features of the SAS LIBNAME statements. We will examine these in order and list some examples and outputs from applying these options.

ENGINE-CONNECTION-OPTIONS

Three options that apply to Excel files are “**HEADER**”, “**MIXED**”, and “**VERSION**”. (The **HEADER** option has two aliases [**GETNAMES** and **HDR**] and the **VERSION** option has one alias [**VER**])

HEADER OPTION

The **HEADER** option impacts the reading of the first line of the Excel file. When the option value is set to **YES** (the default) the first row of data is read to build SAS variable names. The cell values are converted to SAS Variable names. When the text values would generate valid SAS variable names underscores are inserted into the SAS variable name. Duplicate names are avoided by adding a number to the end of the variable name. When the **HEADER** option is set to **NO** or the cell value does not convert to a text field the SAS generates a variable name. Also, when the option is set to **NO** then the first row is considered data, and variable names are generated as F1, F2, F3, ... , to Fn. The following is an example of the output when the **HEADER** option is set to **NO**. Note that columns F4 through F7 were defined as numeric by default, and the titles from row one were translated as missing values.



MIXED OPTION

The **MIXED** Option is provided to assist in reading fields that are not clearly either all text or all numeric. Since Excel files can be generated by someone typing data into one cell after another without regard to the type of data (character or numeric) SAS has to make a default best guess about what the field contains. When an Excel input column contains both character and numeric data this option will cause all of the data in that column to be read in as character data. This option is handled by the Microsoft Jet/Excel engine and is only available in Windows for Excel files. This does not work for delimited files; remember delimited files are read with a FILENAME statement. When this option is set to **YES** the SAS variables are crated as character variables, all numeric data is converted to character data, the Excel file is read in import mode, and no updates are allowed to the file.

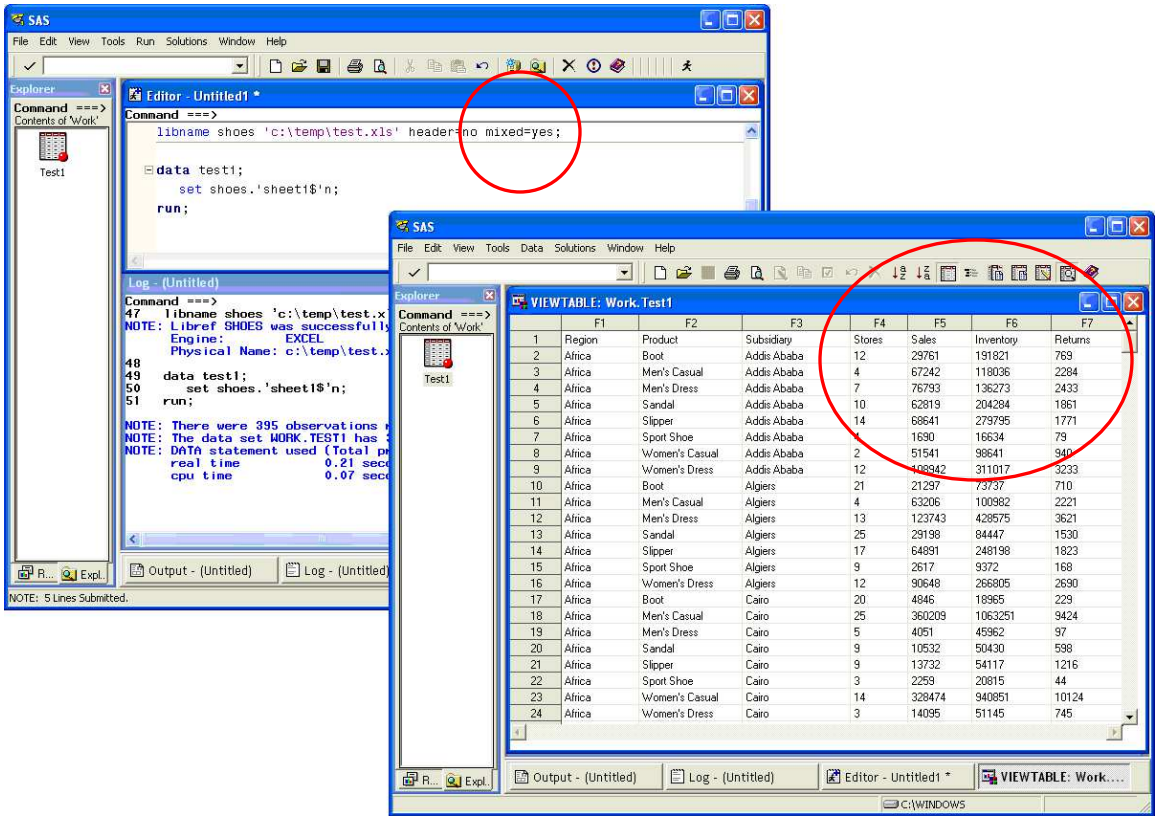
When the **MIXED** option has a value of **NO** (the default value), a specified number of rows of the Excel column are searched and a guess is returned about the data type of the input Excel data field. The Windows Registry can be changed to modify this behavior, but it will change the behavior for all programs that use the Microsoft Jet/Excel engine.

NOTE: This option may cause the wrong variable length to be assigned to the field. The following Windows Registry settings control the behavior of this option, and require administrative privileges to modify. This author does not recommend changing these settings.

[\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Jet\4.0\Engines\Excel]

Values TypeGuessRows and ImportMixedTypes Specific descriptions can be found in the SAS Help files.

An example of the **MIXED** option with a value of **YES** is shown below (combined with **HEADER=NO**):



	F1	F2	F3	F4	F5	F6	F7
1	Region	Product	Subsidiary	Stores	Sales	Inventory	Returns
2	Africa	Boot	Addis Ababa	12	29761	191821	769
3	Africa	Men's Casual	Addis Ababa	4	67242	118036	2284
4	Africa	Men's Dress	Addis Ababa	7	76793	136273	2433
5	Africa	Sandal	Addis Ababa	10	62819	204284	1861
6	Africa	Slipper	Addis Ababa	14	68641	279795	1771
7	Africa	Sport Shoe	Addis Ababa	1	1690	16634	79
8	Africa	Women's Casual	Addis Ababa	2	51541	98641	940
9	Africa	Women's Dress	Addis Ababa	12	408942	311017	3233
10	Africa	Boot	Algiers	21	21297	73737	710
11	Africa	Men's Casual	Algiers	4	63206	100982	2221
12	Africa	Men's Dress	Algiers	13	123743	428575	3621
13	Africa	Sandal	Algiers	25	29198	84447	1530
14	Africa	Slipper	Algiers	17	64891	248198	1823
15	Africa	Sport Shoe	Algiers	9	2617	9372	168
16	Africa	Women's Dress	Algiers	12	90648	266805	2690
17	Africa	Boot	Cairo	20	4846	18965	229
18	Africa	Men's Casual	Cairo	25	360209	1063251	9424
19	Africa	Men's Dress	Cairo	5	4051	45962	97
20	Africa	Sandal	Cairo	9	10532	50430	598
21	Africa	Slipper	Cairo	9	13732	54117	1216
22	Africa	Sport Shoe	Cairo	3	2259	20815	44
23	Africa	Women's Casual	Cairo	14	328474	940851	10124
24	Africa	Women's Dress	Cairo	3	14095	51145	745

There are several changes here, including the loss of the dollar sign in the SAS formatted values, and the presence of the names in row one for columns F4 through F7. In this case only the top row was impacted. However, in Excel files with many values that are mixed types the programmer may need to test each value to determine how to process columns with mixed data. The SAS input function works well to do this type of testing. The following code would work well to split data into two columns (one character and one numeric)

```
LIBNAME xls_data EXCEL 'C:\WUSS_2009\My_Excel_File.xls' HEADER=NO MIXED=YES;

   Data test1;
       Set xls_data.'sheet1'n;
       x = input(F4,10.0);                          * convert to numeric values;
       if x = . then y = input(F4,$char25.);     * if conversion fails, convert;
                                       * to character - y is missing by default;
       drop F4;
   Run;
```

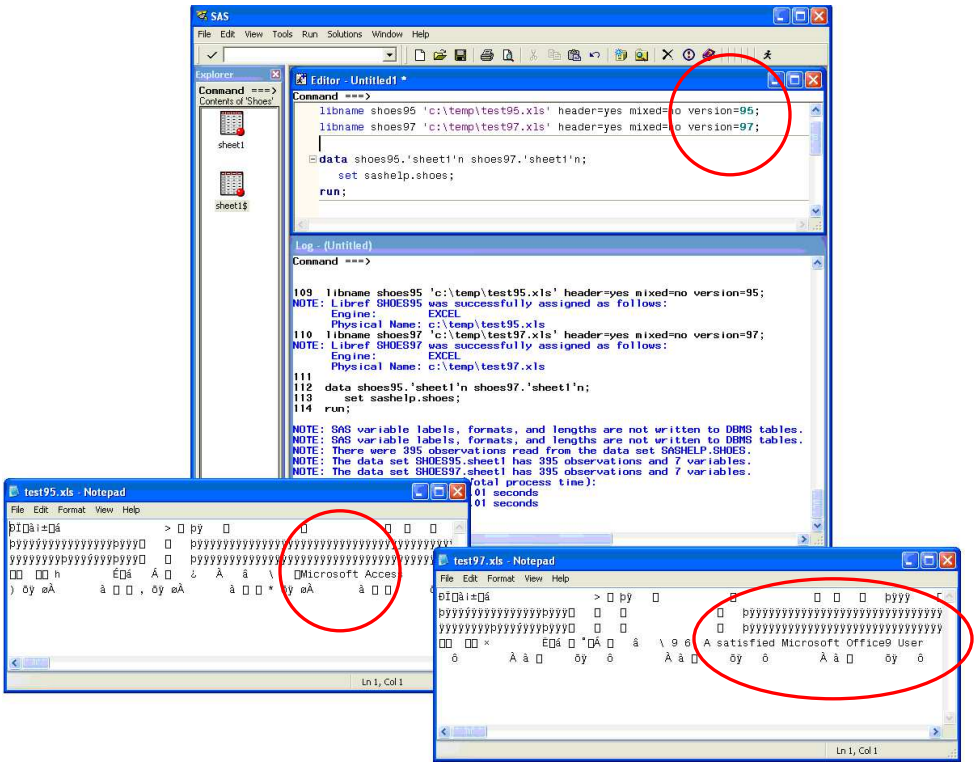
VERSION OPTION

Microsoft Excel has been around for a long time. EXCEL4 was released over 15 years ago, but is not used very much today. While the SAS EXPORT Wizard (with SAS/ACCESS for PC file Formats installed) will process EXCEL4 files the LIBNAME only supports the formats for EXCEL5 (95) EXCEL 97, EXCEL 2000, and EXCEL 2003. (Zender, 2006). SAS LIBNAME support for EXCEL files is in the binary EXCEL (*.xls) format because the Microsoft Jet/Excel engine is used to read and write the files.

The **VERSION** option is not required on input because the Microsoft Jet/Excel engine can determine the format of the input file. However, the output format can be chosen to be one of the two output EXCEL file formats. (EXCEL5, and EXCEL 97 - 2003). The default version is '97'. With the actual option codes listed here '2003', '2002', '2000', '97', '95', '5' (the quotes are valid but not required.)

**** NOTE ** Version 2007 of EXCEL is missing from the list. While EXCEL 2007 can read *.xls files in the native binary formats of 5 through 2003 the native output format for EXCEL 2007 is the *.xlsx file format. This file format is a "ZIP" file. Meaning that it is a file that when the extension (xlsx) is changed to zip, the file can be opened with the WINZIP program and the contents examined. Try it some time, but that is outside the scope of this paper.**

The next example shows both options used to write the output files with each of the files opened with the "NOTEPAD" program to "peek" at the characters inside the files. This will show that the files really do have different formats.



LIBNAME-OPTIONS

The first set of options above applied to the files in their entirety, and was set for all data within the file. This next set of options is not as absolute in the uniform application, and can impact the file or the variables within the file. So here is a list of some of the LIBNAME options that relate to Microsoft Excel, and a brief description of the function they perform. Several will be shown as examples in the space permitted in this paper, and were extracted from the information listed in the SAS Help files for LIBNAME Options for PC Files on Windows. Refer to the Help files for full details about these and other options.

OPTION	DEFAULT	OPTIONS	FUNCTION
ACCESS=	(not set)	READONLY	Prevents write commands if set.
DBGEN_NAME=	DBMS	DBMS/SAS	Specifies how data source column names are created. Option Value DBMS creates SAS names with invalid characters set to "_". Value = "SAS" set names to _COLn where n is the column number starting with zero.
DBMAX_TEXT=	1,024	1 – 32,767	Specifies maximum character string size.
DBSASLABEL=	COMPAT	COMPAT/NONE	Chooses whether or not the source column names are used as SAS label values. This is valid when reading data into a SAS File.
DEFER=	NO	YES/NO	Determines if the input Excel file is opened when the libref is assigned or when the file is first accessed.
DIRECT_SQL=	YES	YES/NO/NONE/ Specific-functionality	This determines if the SQL that SAS generates is passed to the Microsoft Jet/Excel engine when accessing Excel files. (See the SAS Help files for specifics, this Also works with other PC File formats and their respective engines)
SCAN_TEXTSIZE	YES	YES/NO	Scan all Excel Text columns for the longest length of any text filed and use that length for the size of the SAS Variable (The smaller of the text size or the DBMAX_TEXT limit is used as the SAS variable length) Alais – SCAN_TEXT=, SCANTEXT=, and SCANMEMO=
SCAN_TIMETYPE=	NO	YES/NO	Turns on scanning of all rows looking for DATETIME data values and automatically determine what date or time format to apply. An option value of NO assigns either a DATE9. or a DATETIME19. format. See USE_DATETYPE= for more information.
STRINGDATES=	NO	YES/NO	YES reads datetime values as character SAS variables, while NO reads the values into SAS as numeric values.
USE_DATETYPE=	NO	YES/NO	YES uses the DATE format on input and NO uses the DATETIME format while read date values into SAS Variables.

DATASET-OPTIONS

In addition to the LIBNAME options these dataset options are also available when SAS/ACCESS for PC Files is available and the SAS LIBNAME statement is used in conjunction with PC File data. The DROP=, CNTLLEV=, FIRSTOBS=, IN=, KEEP=, RENAME=, and WHERE= dataset options are also valid when using PC data files. Some of these options apply to SAS variables or source data columns individually.

OPTION	FUNCTION
AUTOCOMMIT=	Determines when updates (saves) are written to the PC file in use.
DBCOMMIT=	Saves data after a specified number of rows have been processed.
DBCONDITION=	Defines an SQL selection criteria clause to be passed directly to the Data source for processing (Proc SQL can connect to an Excel file and process data). This SQL clause can be a WHERE, GROUP BY, HAVING, or ORDER BY clause.
DBCREATE_TABLE_OPTS=	Allows the SAS Programmer to append source specific SQL code to be added to a SQL CREATE TABLE command.
DBENCODING=	Defines a different character set to be used in accessing the Excel file.
DBFORCE=	Defines whether or not data values larger than the defined length are inserted into the file, “YES” will insert the value but truncate it. “NO” will cause the value not to be inserted.
DBGEN_NAME=	Determines how to handle creating SAS dataset variable names when the source file has invalid characters in the name.
DBKEY=	Specify a column to use an index for a join using a large and a small dataset, This can improve performance (or hurt it if not used correctly).
DBLABEL=	Choose to use SAS variable labels or names as the output column names (YES = Labels, NO = names)
DBMAX_TEXT=	Define the maximum length of a character or text variable read or written using SAS/ACCESS for PC Files.
DBSASLABEL=	Choose to use the source column names as SAS variable labels
DBSASTYPE=	Specifies a data type that overrides input default variable data type.
DBTYPE=	Specifies a data type that overrides output default variable data type.
INSERT_SQL=	Determines if the data source’s SQL insert method or an alternate method is used to add new rows.
READBUFF=	Determine how many rows are read into the input buffer for processing.
SASDATEFMT=	Define specific date formatting by column when processing dates.

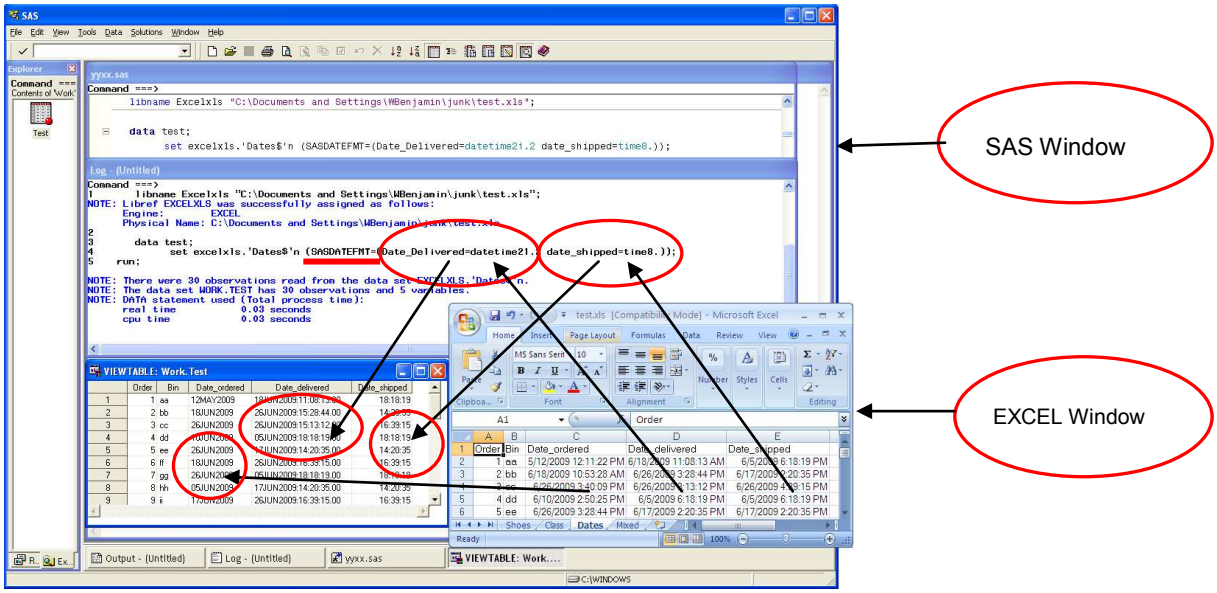
The options selected for this paper represent some of the options that are available for use with Excel files. The specific syntax of these options can be found in the SAS Help files, but has the general form as follows:

```
LIBNAME libref engine-name;
PROC PRINT libref.SAS-data-set-name (OPTION-NAME=option-value);
                                Or
DATA  libref.SAS-data-set-name (OPTION-NAME=option-value);
          Set libref.SAS-data-set-name (OPTION-NAME=option-value);
```

EXAMPLES

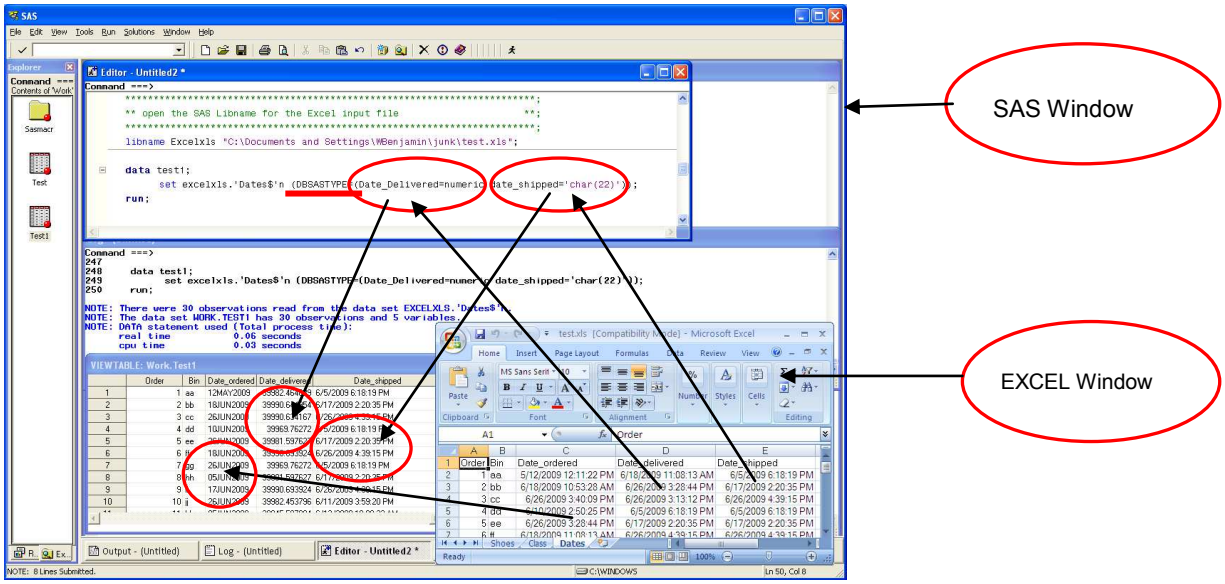
The next pages will show some examples of how some of the dataset options are applied and the results. For the first three, examples an Excel file was created with a numeric “Order” field, a character “Bin” field, and three Excel date/time fields “Date_Ordered”, “Date_Delivered” and “Date_Shipped”. All of the Microsoft date fields are formatted exactly the same way in the Excel workbook.These examples will show how the processing steps required to get data in the format you want, not the SAS default format. There is not enough room for all of the options above to be shown, but several will be illustrated here.

EXAMPLE – 1 – SAS Default date, SAS Datetime21.2, and SAS Time8. Formats.



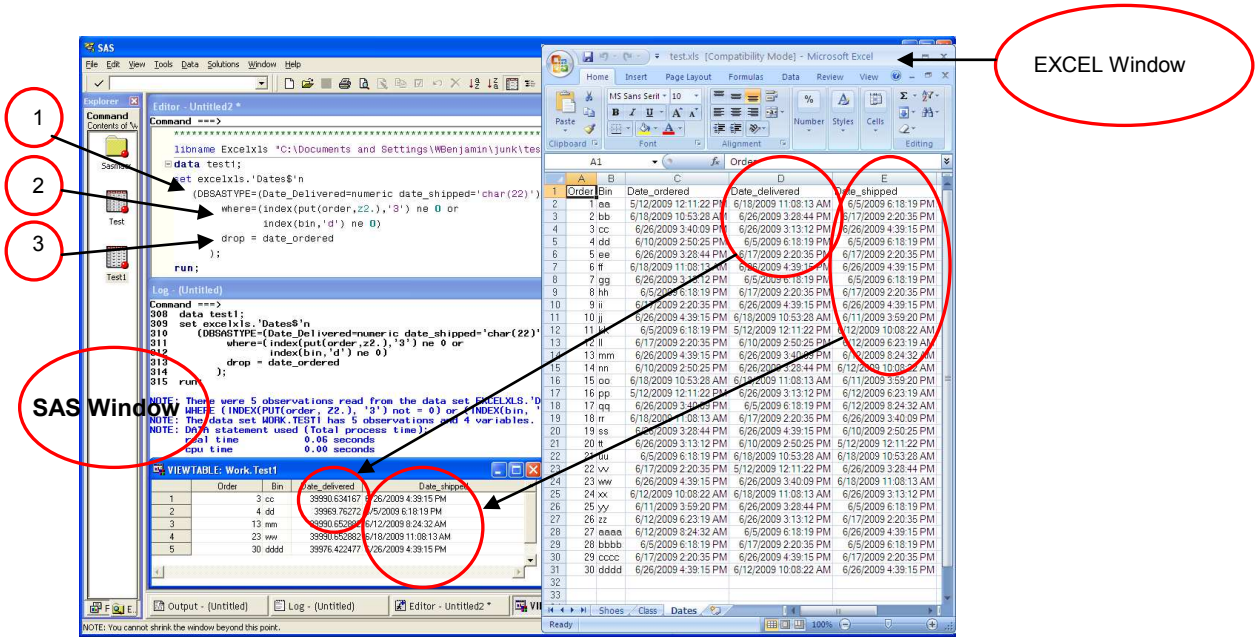
The SASDATEFMT= dataset option controls formatting of dates on input with times being converted to a 24 hour clock instead of “AM” and “PM”.

EXAMPLE – 2 – SAS Default date, SAS numeric field, and SAS character field.



The DBSASTYPE= dataset option controls formatting of fields as numeric or character on input. Note the conversion to an Excel “Data-Time” number/fraction and the retention of the Excel date format in the SAS variable.

EXAMPLE – 3 – Complex WHERE= clause, DROP= clause, and conversion of two fields to a SAS numeric field, and SAS character field.



Complex example.

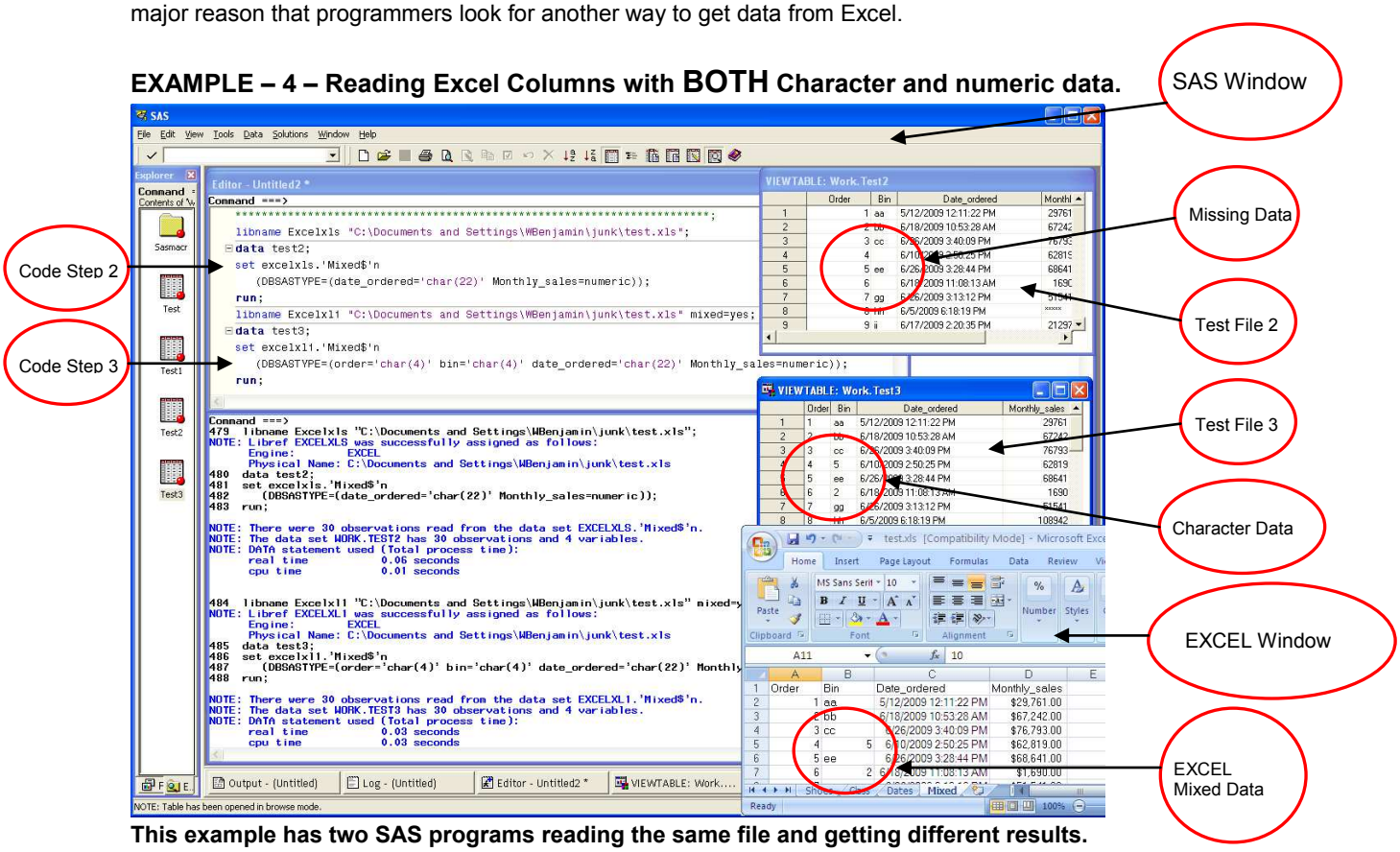
Now let's review these examples. First Example – 1, This SAS code read all five fields from an Excel file, and did nothing with the first three fields (Order, Bin, and Date_Ordered). Then, the SAS SASDATEFMT= dataset option was used to change the Date_Delivered field to a Datetime21.2 format in the SAS file. Finally, the SAS SASDATEFMT= dataset option was used to change the Date_Shipped field to a Time8. format in the SAS file. While this provides enhanced flexibility, it is not a free pass to make any conversion you want. The SAS help files point out that the SAS date format and informat must be equivalent. This restricts the number and type of changes you can make on input from an Excel file. But, you now have control over what changes are made.

Looking at Example – 2 The first three fields are processed the same way as in Example - 1, but the dataset option DBSASTYPE= is used to process the last two date fields. But in this example the Date_Delivered field is converted to a numeric field in the SAS file and the Date_Shipped field is converted to a character field. Anyone who has worked with SAS Date and Date/Time values may notice that the numeric conversion for the Date_Delivered field has HUGE numbers in it as compared to SAS Date and Date/Time values. The reason for the difference is that SAS uses a base date of 01 Jan 1960 and Excel (the Microsoft Jet engine) uses 30 DEC 1899 as the base date.

Example – 3 does the same thing as Example -2, with the addition of WHERE= and DROP= clauses on the SET statement. This example was added to show that the Data Set Options for PC files can be used in addition to other dataset options that are commonly used. (The REPLACE= dataset option is not supported) The picture shown above for Example - 3 includes the full Microsoft Excel file used in the test (a title line and five columns with 30 records), and the full output SAS file that was generated (four variables and five records). Excel rows 4, 14, 24, and 31 were selected because the "Order" column each contained at least one "3". The column contains all numeric values but was converted to a character string to be tested by the WHERE= clause. Neither the input or output values were changed to character fields. Excel rows 5 and 31 also had lowercase "d" values in the input Excel cell. Of course the WHERE= clause only keeps one copy of any row in the input file so row 31 was only selected once. Finally the Date_Ordered field was dropped when the output file was written.

For the next example, an Excel file was created with a numeric “Order” field, a character “Bin” field, a date/time field “Date_Ordered”, and a numeric field formatted as a currency field. This example addresses one of the first things that a SAS programmer notices about reading data from Excel. It may also be the major reason that programmers look for another way to get data from Excel.

EXAMPLE – 4 – Reading Excel Columns with BOTH Character and numeric data.



This example has two SAS programs reading the same file and getting different results.

This final example uses two code segments, each includes a LIBNAME statement and a DATA step. The LIBNAME statements are different because the first one “EXCELXLS” uses no extra options, it just reads the Excel file using the SAS Default options. The second LIBNAME (“EXCELXL1”) uses the SAS LIBNAME option “MIXED=yes”. This option converts EXCEL columns with both character and numeric data into character data when it is being input (from Excel only). The picture above shows the Excel Window with the Bin column of mixed data, SAS File TEST 2 with the Bin variable with missing data, and SAS File TEST 3 with character data in the Bin column and Order variables.

Code Step 2 reads the Excel file and uses the DBSASTYPE= SAS Data Step Option to convert Date_Ordered to character and Monthly_Sales to numeric. But, the code uses the default input routines to read the Order and Bin variables. The result is that rows 4 and 6 have missing values for the Bin variable. The first time a programmer sees this condition and cannot find a way to get the data, is the last time the programmer uses the LIBNAME statement to read Excel data files.

Code Step 3 reads the Excel file and uses the DBSASTYPE= SAS Data Step Option to convert Date_Ordered to character and Monthly_Sales to numeric. It also uses the DBSASTYPE= SAS Data Step Option to convert variable Order to Character (note it is left justified), and variable Bin to character also (note it is left justified too). In this example rows 4 and 6 have the numeric values (converted to character and left justified) in the new SAS dataset (Test3).

Conclusion

The information presented in this paper is intended to show that, like the train in the children's story, the SAS LIBNAME also has an "engine" that won't give up. The SAS/ACCESS for PC Files software is a powerful addition to BASE SAS, and this paper only looked at the options that apply directly to Microsoft Excel files on a PC (except native Microsoft Excel 2007 *.xlsx files). SAS/ACCESS for PC Files will also process Microsoft Access files. These LIBNAME and DATASET options are available to enhance access to non-SAS data files. They are available in DATA steps, and SAS PROC executions, and PROC SQL processing. This software, in conjunction with PC Files Server software, allows these file formats to be processed on LINUX, UNIX, or Windows – 64 Bit platforms via a network connection. This pushes the power of the LIBNAME engine far beyond the IMPORT/EXPORT engines of Base SAS.

This paper was written for all of the programmers that have ask "Why can't SAS read and write my Excel files better than that?" The paper was written to show them that SAS can do better, if only the programmer takes the time to just stop using the defaults, and find out what the SAS LIBNAME Engine really can do for them!

Bibliography

Zender, Cynthia, 2006, Base SAS® to Microsoft Excel: Counting the Ways, SAS Institute Inc Cary, NC.

SAS Help files distributed with the Base SAS system and SAS/ACCESS for PC Files were referenced above.

CONTACT INFORMATION

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

Name	William E Benjamin Jr
Enterprise	Owl Computer Consultancy, LLC
Address	P.O. Box 42434
City, State ZIP	Phoenix AZ, 85023
Work Phone:	602-942-0370
Fax:	602-942-3204
E-mail:	William@OwlComputerConsultancy.com