CHAPTER

10

XLS Chapter

Note to UNIX and OS/390 Users

If you are running this SAS/ACCESS interface under the UNIX or OS/390 operating environment, this chapter does not apply to you. Instead, see Chapter 3, "DBF and DIF Procedures," on page 33. Under UNIX and PC hosts, you can use these procedures to convert a DBF or DIF file to a SAS data set or a SAS data set to a DBF or DIF file. Under OS/390, you can use PROC DBF *only* to convert a DBF file to a SAS data set or a SAS data set to a DBF file.

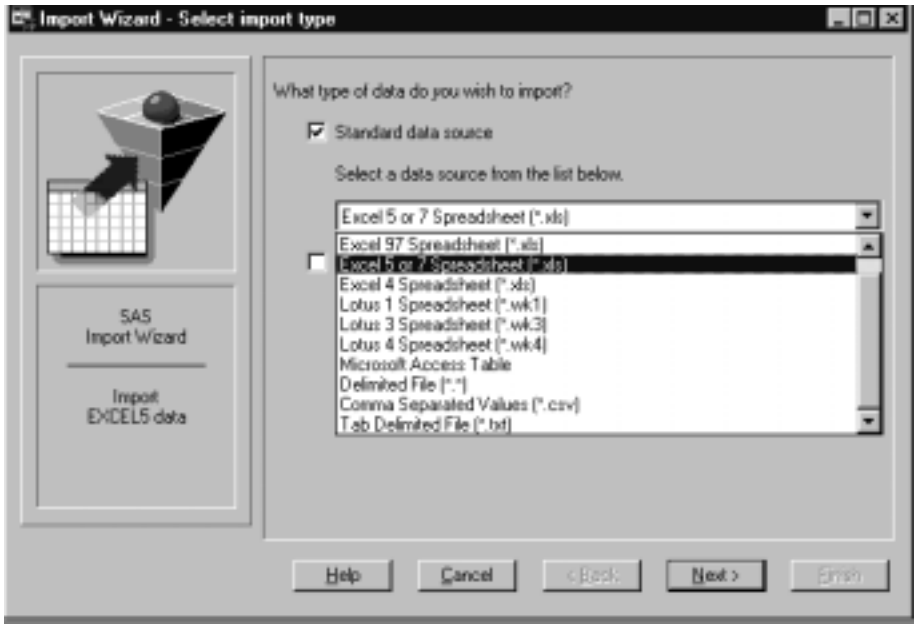
Import/Export Facility

UNIX and PC users can access Excel (or XLS) data—including Excel 97 files—through the Import/Export facility or by using the IMPORT and EXPORT procedures. An overview is included in Chapter 5, "Import/Export Facility and Procedures," on page 51.

To use the point-and-click interface from a SAS PROGRAM EDITOR window, select the **File** menu and then select the **Import Data** or **Export Data** item. Information

about how to import or export Excel data is available from the Help button. The following is a sample Import window:

Display 10.1 Import Window



To write code to import or export Excel data, refer to the detailed descriptions of the IMPORT and EXPORT procedures in the *SAS Procedures Guide*. This documentation also includes several examples.

Understanding XLS Essentials

This chapter introduces SAS System users to XLS files. It focuses on the terms and concepts that help you use the SAS/ACCESS interface and includes descriptions of

- XLS files
- XLS file naming conventions
- XLS file data types
- how the SAS/ACCESS interface works.

XLS files contain data in the form of Microsoft Excel spreadsheets. Unless otherwise noted, Excel 4 and Excel 5 files are referred to collectively throughout this report as XLS. Excel 5 files are not supported under OS/2.

Note: The ACCESS and DBLOAD procedures do not support files in the Excel 97 (Version 8) format. However, you can still access Excel 97 files through the SAS Import/Export facility. From a SAS session's PROGRAM EDITOR window, select the **File** menu and then select the **Import Data** or **Export Data** item. Information about how to import and export Excel data is available from the Help button. To write code to import or export Excel data, refer to the IMPORT or EXPORT procedure description in the *SAS Procedures Guide*. △

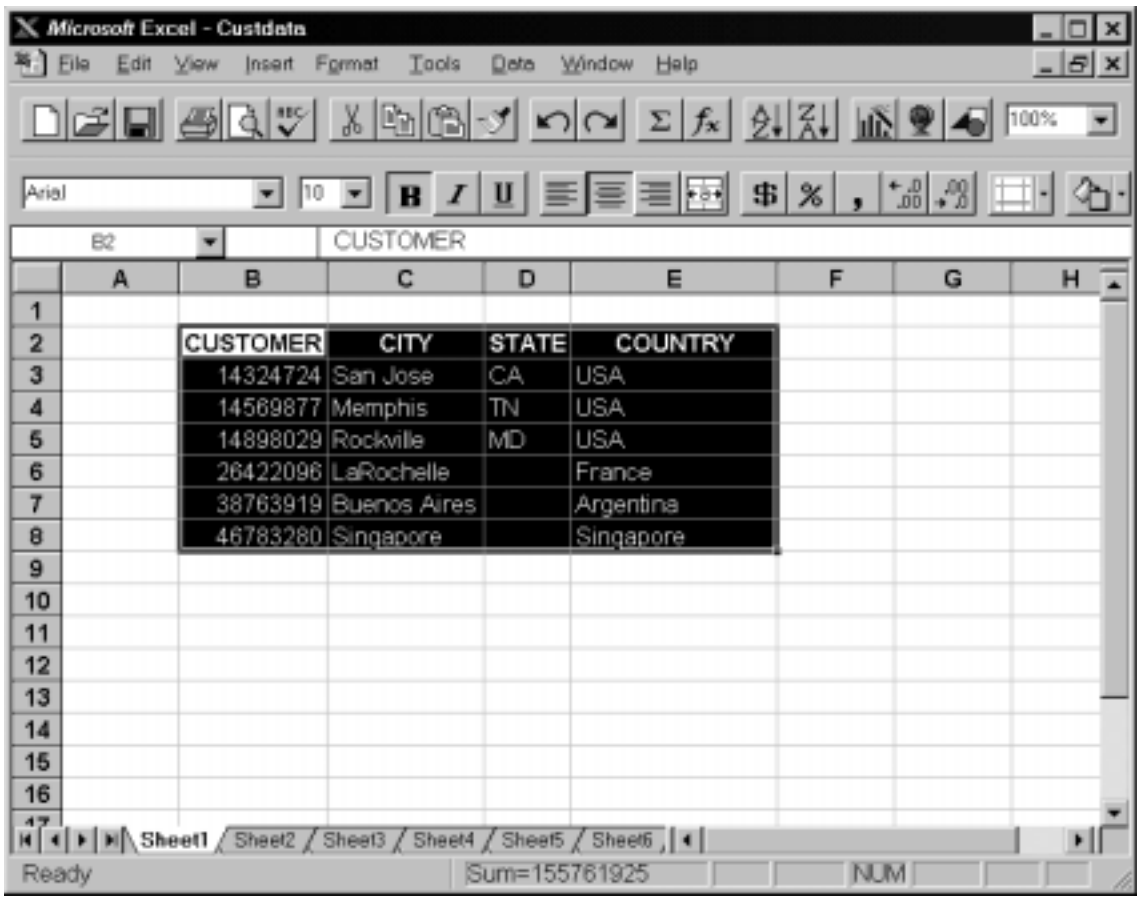
XLS Files

Various software products, such as the Microsoft Excel spreadsheet, enable you to use spreadsheet or database files to enter, organize, and perform calculations on data. Spreadsheets are most often used for general ledgers, income statements, and other types of financial record keeping. Database files also enable you to organize related information, such as the data in an accounts-receivable journal.

In spreadsheets, the data are organized according to certain relationships among data items. These relationships are expressed in a tabular form—in columns and rows. Each *column* represents one category of data, and each *row* can hold one data value for each column.

A Microsoft Excel 5.0 worksheet, for example, is an electronic spreadsheet consisting of a grid of 256 columns and 16,384 rows. The intersection of a column and a row is called a *cell*. Display 10.2 on page 133 illustrates a portion of a standard Excel worksheet.

Display 10.2 Columns and Rows of Data in an XLS File



Column letters for each column appear above the worksheet. Columns are lettered A through IV (A to Z, AA to AZ, BA to BZ, and so on to IV). Row numbers for each row appear to the left of the worksheet. Rows are numbered 1 to 16,384. For Excel 4 files, only one worksheet (worksheet 1) is allowed per file, but more than one worksheet can be stored in a workbook. You must convert any worksheets you store in a workbook back to worksheets before you can use the data in a SAS program.

A *range* is a subset of cells in a worksheet. A range is identified by its address, which begins with the name of the top left cell and ends with the name of the bottom right cell separated by two periods. For example, the range B3..D6 is the range address for a rectangular block of 12 cells whose top left cell is B3 and whose bottom right cell is D6 (as shaded in the display).

XLS File Naming Conventions

The following conventions apply to XLS filenames. Filenames must also follow operating-system specific conventions, so check the documentation that comes with your Microsoft Excel product or other software products for further information.

- Under Windows 95, Windows 98, Windows NT, and OS/2, the ACCESS and DBLOAD procedures support long names that are specified in the PATH= statement (such as **`path= 'c:\sasdemo\library\new_customer_1999.xls';`**). However, XLS files with long names might not accepted by some versions of Microsoft Excel.
- Filenames start with a letter, and they can contain any combination of the letters A through Z, the digits 0 through 9, the underscore (_), the hyphen (-), and spaces (blanks) within filenames.
- Filenames can contain spaces. Filenames that contain spaces or lowercase letters are supported by the ACCESS and DBLOAD procedures, but they might not be accepted by some versions of Microsoft Excel.

XLS Data Types

Microsoft Excel software has two data types: character and numeric. Microsoft Excel character data may be entered as labels or formula strings; Microsoft Excel numeric data may be entered as numbers or formulas.

Character data are generally considered text and can include dates and numbers.

Numeric data can include numbers (0 through 9), formulas, and cell entries that begin with one of the following symbols: +, $, @, –, =, or #. When you create and load an Excel file with PROC DBLOAD, the SAS/ACCESS engine supplies **`#NA`** for a missing, numeric value.

Numeric data also can include date and time values. In Microsoft Excel software, a *date value* is the integer portion of a number that can range from 01 January 1900 to 31 December 2078, that is, 1 to 65380. A Microsoft Excel software *time value* is the decimal portion of a number that represents time as a proportion of a day. For example, 0.0 is midnight, 0.5 is noon, and 0.999988 is 23:59:59 (on a 24-hour clock). While a number can have both a date and a time portion, the formats in Microsoft Excel display a number only in a date, time, or datetime format. For information on how the SAS/ACCESS interface handles date and time values and formats, see “How the SAS System Handles Date and Time Values” on page 135.

When you create an access descriptor, the interface software uses the column types and formats in the XLS file to determine the corresponding SAS variable formats. The SAS System generates its default formats based on the values that you specify for the SCANTYPE, SKIPROWS, and GETNAMES statements (or in the corresponding fields in the Access Descriptor Identification window). You can change the formats generated by the software interface. For more information, see “How the SAS/ACCESS Interface Works” on page 147.

When you create an access descriptor, any data value that does not match the column type (character or numeric) is treated as a missing value. This is the default action. However, you can use the MIXED=YES statement to convert numeric data values in a character column to their character representation.

You can also set the SS_MIXED environment variable to **YES** in your SAS configuration file so that both numeric and character data are displayed as SAS character data. Add this line to your SAS configuration file:

```
-SET SS_MIXED YES
```

See “Setting Environment Variables” on page 142 for more information on environment variables. For more information on changing the column type from the type determined by SAS/ACCESS software when you create an access descriptor, refer to the sections on XLS-specific procedure statements later in this chapter.

How the SAS System Handles Date and Time Values

The conversion of date and time values between SAS data sets and Microsoft Excel spreadsheets is transparent to users. However, you are encouraged to understand the differences between them.

Microsoft Excel date and time values and formats are described in “XLS Data Types” on page 134.

Datetime Conversions in the ACCESS Procedure

As described earlier in this chapter, an XLS date value is the integer portion of a number that represents the number of days between January 1, 1900 and a specified date. An XLS time value is a decimal portion of a number that represents time as a portion of the day. For example, 0.0 is 12:00:00 a.m., and 0.9999884 is 11:59:59 p.m. While a number can have both a date and a time portion, the formats in XLS display a number only in a date format or in a time format. For example, for 1:00 p.m., March 12, 1994, the XLS date value is 34405, the time value is 0.5416667, and the datetime value is 34405.5416667.*

The SAS System handles date and time values differently than XLS. A SAS date value is an integer that represents the number of days between January 1, 1960 and a specified date. A SAS time value is an integer that represents the number of seconds since midnight of the current day. When a date and a time are both present, the SAS System stores the value as the number of seconds since midnight, January 1, 1960. For example, for 1:00 p.m., March 12, 1994, the SAS date value is12489, and the SAS time value is 46800. Therefore, the SAS datetime value is 1079096400.

When you create an access descriptor, the SAS System converts an XLS datetime format to its corresponding SAS datetime format if an XLS datetime format is specified for the variable in the XLS file. Note that if the datetime value does not have an XLS format in the XLS file, the SAS System treats the datetime value like a numeric value.

To convert an XLS datetime format to a SAS datetime format, you need a SAS datetime format in the access descriptor. For example, changing the default SAS numeric format (15.2) to a SAS date format in the descriptor causes the XLS date value (based on January 1, 1900) to be converted to an equivalent SAS date value (based on January 1, 1960). In other words, the XLS numeric value for January 1, 1960 (which is 21916) is converted to the equivalent SAS representation of January 1, 1960 (which is 0) only if a SAS datetime format is assigned in the descriptor for that column. Otherwise, the XLS value of 21916 is treated as a SAS numeric value of 21916.

The following table shows how the SAS System uses a Microsoft Excel datetime value to convert to a SAS datetime format.

* In this description, datetime (in lowercase) refers to any value or format that represents a date, a time, or both a date and a time.

Table 10.1 Value-to-Format Conversions

For a SAS format	SAS System uses
date	integer portion of the Microsoft Excel number
time	decimal portion of the Microsoft Excel number
date-and-time	integer and decimal portion of the Microsoft Excel number

Datetime Conversions in the DBLOAD Procedure

If a SAS variable is specified with a date, time, or datetime format in the FORMAT statement, the interface view engine converts that SAS datetime format into the equivalent Microsoft Excel datetime format when the new XLS file is created.

However, if a SAS datetime format is not specified in the input SAS data set, you have to assign a format by using a PROC DBLOAD FORMAT statement. Doing so assigns a Microsoft Excel datetime format to the SAS variable when the variable is loaded into an XLS file. If you do not assign a SAS datetime format, the SAS numeric value for the date is written to the XLS file. Because SAS dates are based on January 1, 1960, and Microsoft Excel dates are based on January 1, 1900, the date value in the XLS file will be inaccurate.

To maintain a SAS variable format in the input data set, yet change it just while the DBLOAD procedure is in progress, use the FORMAT statement in PROC DBLOAD. This statement enables you to assign a temporary format to a SAS variable for the duration of the procedure without affecting the input SAS data set.

For example, if the SAS format for the BIRTHDAT variable in the MYDATA.SASEMPS access descriptor is left at the default 15.2 format, you can specify the FORMAT statement to change the variable's format to DATE7. while you are creating and loading the XLS file. When you load the XLS file, the DATE7. format becomes an equivalent Microsoft column format, DDMMMYY. When the DBLOAD procedure has completed, the SAS format for the BIRTHDAT variable returns to the 15.2 format.

You can specify the FORMAT statement in the PROC DBLOAD statement when you invoke the procedure using any of the methods of processing.

ACCESS Procedure Data Conversions

Use PROC ACCESS to define descriptors that identify spreadsheet data and the conversions necessary to use the data in SAS programs. The Microsoft Excel label data type is formatted as a SAS character type, and the Microsoft Excel number data type is formatted as a SAS numeric type.

Fonts, attributes, and colors in the XLS files are not read into the SAS data sets. However, the ACCESS procedure supports most of the XLS number formats and automatically converts them to the corresponding SAS formats. Any XLS data strings longer than 200 characters are truncated while being converted into SAS data sets, and any SAS data file created from XLS files can contain up to 256 variables and 16,384 observations.

Table 10.2 on page 137 shows the default SAS System variable formats that the ACCESS procedure assigns to each type of standard XLS file data. Table 10.3 on page 139 provides SAS System variable formats for customized XLS format strings. XLS file numeric data include date and time values. See “How the SAS System Handles Date and Time Values” on page 135 for more information.

Table 10.2 Default SAS System Variable Formats for XLS File Data

XLS File Data		SAS Variable Format	
Data Type	XLS Format String	Type	Format
Char1	@2	Char	$*w*.
Numeric3	General	Num	BEST
Numeric	0	Num	*w.d*
Numeric	0.00	Num	*w.d*
Numeric	#,##0	Num	COMMA*w.d*
Numeric	#,##0.00	Num	COMMA*w.d*
Numeric	#,##0_);(#,##0)	Num	NEGPAREN*w.d*
Numeric	#,##0_);[Red](#,##0)	Num	NEGPAREN*w.d*
Numeric	#,##0.00_);(#,##0.00)	Num	NEGPAREN*w.d*
Numeric	#,##0.00_);[Red](#,##0.00)	Num	NEGPAREN*w.d*
Numeric	$#,##0_);($#,##0)	Num	DOLLAR*w.d*
Numeric	$#,##0_);[Red]($#,##0)	Num	DOLLAR*w.d*
Numeric	($#,##0.00_);($#,##0.00)	Num	DOLLAR*w.d*
Numeric	($#,##0.00_);[Red]($#,##0.00)	Num	DOLLAR*w.d*
Numeric	_($*#,##0_);_($*(#,##0);_($*"-"_);_(@_)	Num	DOLLAR*w.d*
Numeric	_(*#,##0_);_(*(#,##0);_(*"-"_);_(@_)	Num	NEGPAREN*w.d*
Numeric	_($*#,##0.00_);_($*(#,##0.00);_($*"??_);_(@_)	Num	DOLLAR*w.d*
Numeric	_(*#,##0.00_);_(*(#,##0.00);_(*"-"??_);_(@_)	Num	NEGPAREN*w.d*
Numeric	0%	Num	PERCENT*w.d*
Numeric	0.00%	Num	PERCENT*w.d*
Numeric	0.00E+00	Num	E*w.d*
Numeric	##0.0E+0	Num	E*w.d*
Numeric	*m/d/yy*	Num	MMDDYY*w.*
Numeric	*d-mmm-yy*	Num	MMDDYY*w.*
Numeric	*d-mmm*	Num	DATE*w.*
Numeric	*mmm-yy*	Num	MONYY*w.*
Numeric	*h:mm* AM/PM	Num	TIME*w.*
Numeric	*h:mm:ss* AM/PM	Num	TIME*w.*
Numeric	*h:mm*	Num	TIME*w.*
Numeric	*hh:mm*	Num	TIME*w.*
Numeric	*h:mm:ss*	Num	TIME*w.*

XLS File Data		SAS Variable Format	
Data Type	XLS Format String	Type	Format
Numeric	*hh:mm:ss*	Num	TIME*w.*
Numeric	*m/d/yy h:mm*	Num	DATETIME*w.*
Numeric	*ddmmmyy*	Num	DATE*w.*
Numeric	*ddmmmyyyy:hh:mm:ss*	Num	DATETIME*w.*
Numeric	*dd*	Num	DATE*w.*
Numeric	*dd/mm/yy*	Num	DDMMYY*w.*
Numeric	*dddd*	Num	DATE*w.*
Numeric	*mm/dd/yy*	Num	MMDDYY*w.*
Numeric	*mm:ss*	Num	MMSS*w.*
Numeric	*mm yy*	Num	MONYY*w.*
Numeric	*mm yyyy*	Num	MONYY*w.*
Numeric	*mm:yy*	Num	MONYY*w.*
Numeric	*mm:yyyy*	Num	MONYY*w.*
Numeric	*mm-yy*	Num	MONYY*w.*
Numeric	*mm-yyyy*	Num	MONYY*w.*
Numeric	*mmyy*	Num	MONYY*w.*
Numeric	*mmyyyy*	Num	MONYY*w.*
Numeric	*mm.yy*	Num	MONYY*w.*
Numeric	*mm.yyyy*	Num	MONYY*w.*
Numeric	*mm/yy*	Num	MONYY*w.*
Numeric	*mm/yyyy*	Num	MONYY*w.*
Numeric	*mmmm*	Num	MONYY*w.*
Numeric	*m*	Num	MONYY*w.*
Numeric	*mmmyy*	Num	MONYY*w.*
Numeric	*mmmyyyy*	Num	MONYY*w.*
Numeric	*dddd, mmmm dd, yyyy*	Num	MONYY*w.*
Numeric	*dddd, dd mmmm yyyy*	Num	MONYY*w.*
Numeric	*mmmm dd, yyyy*	Num	MONYY*w.*
Numeric	*dd mmmm yyyy*	Num	MONYY*w.*
Numeric	*yy*	Num	YYMMDD*w.*
Numeric	*yyyy*	Num	YYMMDD*w.*
Numeric	*yy mm*	Num	YYMMDD*w.*
Numeric	*yyyy mm*	Num	YYMMDD*w.*
Numeric	*yy:mm*	Num	YYMMDD*w.*
Numeric	*yyyy:mm*	Num	YYMMDD*w.*
Numeric	*yy-mm*	Num	YYMMDD*w.*

XLS File Data		SAS Variable Format	
Data Type	XLS Format String	Type	Format
Numeric	*yyyy-mm*	Num	YYMMDD*w.*
Numeric	*yymm*	Num	YYMMDD*w.*
Numeric	*yyyymm*	Num	YYMMDD*w.*
Numeric	*yy.mm*	Num	YYMMDD*w.*
Numeric	*yyyy.mm*	Num	YYMMDD*w.*
Numeric	*yy/mm*	Num	YYMMDD*w.*
Numeric	*yyyy/mm*	Num	YYMMDD*w.*
Numeric	*yy-mm-dd*	Num	YYMMDD*w.*
Numeric	*yymmm*	Num	YYMMDD*w.*
Numeric	*yyyymmm*	Num	YYMMDD*w.*

1 Label data.
2 The XLS character format for Excel Version 5.0.
3 Number, formula, or missing data.

Table 10.3 Default SAS System Variable Formats for Customized XLS Format Strings

XLS File Data		SAS Variable Format	
Data Type	XLS Format String	Type	Format
Numeric	"$"	Num	DOLLAR*w.d*
Numeric	"E"	Num	E*w.d*
Numeric	"*m*, *d* and *y*"	Num	MMDDYY*w.*
Numeric	"*m* and *h*"	Num	TIME*w.d*
Numeric	"*m* and *s*"	Num	TIME*w.d*
Numeric	"*m* and *y*"	Num	MONYY*w.*
Numeric	"*m*"	Num	DATE*w.*
Numeric	"*d*"	Num	DATE*w.*
Numeric	"*y*"	Num	DATE*w.*
Numeric	"0.0"	Num	*w.d*
Numeric	Fraction values (#?/?)	Num	BEST*w.d*
Numeric	Percent values (0.0%)	Num	PERCENT*w.d*
Numeric	All others	Num	BEST*w.d*

Note that *w* is based on Excel column width; *.d* is controlled by the Excel format string.

If XLS file data fall outside of the valid SAS data ranges, you receive an error message in the SAS log when you try to access the data.

The SAS/ACCESS interface does not fully support the Microsoft Excel hidden and text formats. XLS data in hidden format are displayed in SAS data sets; however, you can drop the hidden column when you are creating the access descriptor. If you want to display the formula in the text format, add a space to indicate that the formula entry is a label. Otherwise, the results of the formula are displayed.

If you have set the SS_MIXED environment variable to YES, the numerical values in XLS files are converted to character strings in SAS data sets if the corresponding SAS variable type is specified as character.

DBLOAD Procedure Data Conversions

This section explains how SAS data are read into Microsoft Excel data when a table is loaded. In this conversion, the SAS character data type is converted into the Microsoft Excel label type and the SAS numeric type is converted into the Microsoft Excel number type.

The SAS/ACCESS interface automatically converts SAS formats to the same or associated Microsoft Excel formats and column widths. However, you can temporarily assign other formats and column widths to SAS variables by using the FORMAT statement so that the loaded XLS file columns have the formats you want. Table 10.4 on page 140 shows the SAS System variable types and formats and the XLS data types, formats, and column widths that you can assign them to.

Note: The FORMAT statement in PROC DBLOAD only changes the format of SAS variables while you are creating and loading the XLS files. When the procedure is completed, the formats of SAS variables return to their original settings. △

XLS values are numeric data. See "How the SAS System Handles Date and Time Values" on page 135 for more information.

Table 10.4 Converting SAS System Variable Formats to XLS File Data

SAS Variable Format		XLS File Data	
Type	Format	XLS Format String	Data Type
Char	" "	General	LABEL
Char	$CHAR	General	LABEL
Char	$	General	LABEL
Num	BEST*w.d*	General	NUMBER
Num	COMMA*w.d*	#,##0	NUMBER
Num	COMMAX*w.d*	#,##0	NUMBER
Num	DATE*w.*	*ddmmmyy*	NUMBER
Num	DATETIME*w.d*	*ddmmmyyyy*:*hh*:*mm*:*ss*	NUMBER
Num	DAY*w.*	dd	NUMBER
Num	DDMMYY*w.*	*dd*/*mm*/*yy*	NUMBER
Num	DOLLAR*w.d*	"$"#,##0_);("$"#,##0)	NUMBER
Num	DOLLARX*w.d*	"$"#,##0_);("$"#,##0)	NUMBER
Num	DOWNAME*w.d*	*dddd*	NUMBER
Num	E*w.*	0.00E+00	NUMBER
Num	HHMM*w.d*	*h*:*mm*	NUMBER
Num	HOUR*w.d*	*h*:*mm*	NUMBER
Num	JULDAY*w.*	*m*/*d*/*yy*	NUMBER

SAS Variable Format		XLS File Data	
Type	Format	XLS Format String	Data Type
Num	JULIAN*w*.	*m*/*d*/*yy*	NUMBER
Num	MMDDYY*w*.	*mm*/*dd*/*yy*	NUMBER
Num	MMSS*w*.*d*	*mm*:*ss*	NUMBER
Num	MMYY*x**w*.	*mm yy*	NUMBER
Num	MMYYC	*mm*:*yy*	NUMBER
Num	MMYYD	*mm*-*yy*	NUMBER
Num	MMYYN	*mmyy*	NUMBER
Num	MMYYP	*mm*.*yy*	NUMBER
Num	MMYYS	*mm*/*yy*	NUMBER
Num	MONNAME*w*.	*mmmm*	NUMBER
Num	MONTH*w*.	*m*	NUMBER
Num	MONYY*w*.	*mmmyy*	NUMBER
Num	NEGPAREN*w*.*d*	#,##0_);(#,##0)	NUMBER
Num	NENGO*w*.	*m*/*d*/*yy*	NUMBER
Num	PERCENT*w*.*d*	0%	NUMBER
Num	QTR*w*.	*m*/*d*/*yy*	NUMBER
Num	QTRR*w*.	*m*/*d*/*yy*	NUMBER
Num	SSN*w*.	000-00-0000	NUMBER
Num	TIME*w*.*d*	*h*:*mm*:*ss*	NUMBER
Num	TOD*w*.	*h*:*mm*:*ss*	NUMBER
Num	W	0	NUMBER
Num	WEEKDATE*w*.	*dddd*, *mmmm dd*, *yyyy*	NUMBER
Num	WEEKDATX*w*.	*dddd*, *dd mmmm yyyy*	NUMBER
Num	WEEKDAY*w*.	m/d/yy	NUMBER
Num	WORDDATE*w*.	*mmmmdd*, *yyyy*	NUMBER
Num	WORDDATX*w*.	*dd mmmm yyyy*	NUMBER
Num	YEAR*w*.	*yy* or *yyyy*	NUMBER
Num	YYMM	*yy mm*	NUMBER
Num	YYMMC	*yy*:*mm*	NUMBER
Num	YYMMD	*yy*-*mm*	NUMBER
Num	YYMMN	*yymm*	NUMBER
Num	YYMMP	*yy*.*mm*	NUMBER
Num	YYMMS	*yy*/*mm*	NUMBER
Num	YYMMDD*w*.	*yy*-*mm*-*dd*	NUMBER
Num	YYMON*w*.	*yymmm*	NUMBER

SAS Variable Format		XLS File Data	
Type	Format	XLS Format String	Data Type
Num	Z*w.d*	0*w.d*	NUMBER
Num	FRACT*w.*	# ?/?	NUMBER

Note that Excel column widths are set to *w* and displayed in the column. If data are larger than column width, the data are displayed as pound signs (###), in which case the data can be viewed by adjusting the column width.

Setting Environment Variables

You can change the default behavior of the SAS/ACCESS interface by setting environment variables in your SAS configuration file. You can set three SAS/ACCESS environment variables: SS_MIXED, SS_NAMES, and SS_SCAN. Setting these variables in your SAS configuration file changes how the interface works by default .

The configuration file omits the following environment variables. When the environment variables are omitted, the default value for them is NO.

SS_MIXED YES | NO

YES allows both Microsoft Excel numeric and character data in a column to be displayed as SAS character data. The Microsoft Excel numeric data are converted to their character representation when their corresponding SAS variable type is defined as character.

NO does not convert Microsoft Excel numeric data in a column into SAS character data. Microsoft Excel numeric data are read in as SAS missing values when their corresponding SAS variable type is defined as character. NO is the default.

Setting the SS_MIXED environment variable changes the default value of the MIXED statement in PROC ACCESS.

SS_NAMES YES | NO

YES in PROC ACCESS generates SAS variable names from column names in the first row of the worksheet or the specified range of the worksheet and reads data from the second row. YES in PROC DBLOAD writes column names using SAS variable names or SAS variable labels to the first row of the new XLS file, reads data from the data set, and writes them to the XLS file beginning with the second row.

NO in PROC ACCESS generates the SAS variable names VAR0, VAR1, VAR2, and so on, and reads data from the first row of the worksheet or specified range. NO in PROC DBLOAD reads the data from the data set and writes them to the XLS file beginning with the first row. NO is the default.

Setting the SS_NAMES environment variable changes the default value of the GETNAMES statement in PROC ACCESS and the PUTNAMES statement in PROC DBLOAD.

SS_SCAN YES | NO | *number-of-rows*

YES scans the data type and format of rows in a worksheet or specified range after skipping the number of rows specified in the SKIPROWS statement. SS_SCAN finds the most common Microsoft Excel data type and format in order to

generate the default SAS data type and format. If a number of rows is specified, SAS/ACCESS software scans only the data type and format from these rows.

NO uses the type and format of the first row in a worksheet or specified range after skipping the number of rows specified in SKIPROWS to generate the default SAS data type and format. NO is the default.

Number-of-rows scans the type and format of the specified number of rows only. Setting the number of rows is more efficient because data are read only from the specified number of rows rather than from the entire file.

Setting the SS_SCAN environment variable changes the default value of the SCANTYPE statement in PROC ACCESS.

ACCESS Procedure: XLS Specifics

Chapter 2, "ACCESS Procedure Reference," on page 11 describes the generic options and procedure statements that enable you to create access descriptors, view descriptors, and SAS data files from PC file format data. The following section describes the PC file-specific statements you use in the SAS/ACCESS interface to XLS data.

ACCESS Procedure Statements for XLS

To create an access descriptor, you use the DBMS=XLS option and six database-description statements: PATH=, GETNAMES, RANGE, SCANTYPE, SKIPROWS, and WORKSHEET. These database-description statements supply XLS-specific information to the SAS System, and must immediately follow the CREATE statement that specifies the access descriptor to be created. In addition to the database-description statements, you can use editing statements when you create an access descriptor. These editing statements must follow the database-description statements.

Database-description statements are only required when you create access descriptors. Because XLS information is stored in an access descriptor, you do not need to repeat this information when you create view descriptors.

The SAS/ACCESS interface to XLS uses the following procedure statements in batch mode:

PROC ACCESS DBMS=XLS | EXCEL;
CREATE *libref.member-name.*ACCESS | VIEW;
UPDATE *libref.member-name.*ACCESS | VIEW;

GETNAMES <=> YES | NO | Y | N;
PATH= *'path-and-filename'*<.XLS> '| <'>*filename*<'> | *fileref*;
RANGE <=> <'>*range-name*<'> | *'range-address'*;
SCANTYPE <=> YES | NO | Y | N | <*number-of-rows*>;
SKIPROWS <=> *number-of-rows-to-skip*;
WORKSHEET <=> *worksheet-name*;
ASSIGN <=> YES | NO | Y | N ;
DROP <'>*column-identifier-1*<'> <...<'>*column-identifier-n*<'>>;
FORMAT <'>*column-identifier-1*<'> <=> *SAS-format-name-1*
<...<'>*column-identifier-n*<'> <=> *SAS-format-name-n*> ;
LIST <ALL | VIEW | <'>*column-identifier*<'>> ;

MIXED <=> YES | NO | Y | N;
RENAME <'>*column-identifier-1*<'> <=> *SAS-variable-name-1*
<...<'>*column-identifier-n*<'> <=> *SAS-variable-name-n*> ;
RESET ALL | <'>*column-identifier-1*<'> <...<'>*column-identifier-n*<'>> ;
SELECT ALL | <'>*column-identifier-1*<'> <...<'>*column-identifier-n*<'>> ;
SUBSET selection-criteria ;
TYPE *column-identifier-1* <=> C | N <... *column-identifier-n* <=> C | N>;
UNIQUE <=> YES | NO | Y | N ;
RUN;

Note: By default, PROC ACCESS uses Excel 5.0. Excel 5.0 files have the identical format to Excel 95 (Version 7) files. △

The QUIT statement is also available in PROC ACCESS. However, its use causes the procedure to terminate. QUIT is used most often in the interactive line and noninteractive modes to exit the procedure without exiting SAS.

GETNAMES <=> YES | NO | Y | N;
determines whether SAS variable names are generated from column names in the first row of the range when an access descriptor is created. When you update a descriptor, you are not allowed to specify the GETNAMES statement.

The GETNAMES statement is optional. If you omit it, the default value GETNAMES=NO is used, and the XLS interface generates the SAS variable names VAR0, VAR1, VAR2, and so on. If you specify GETNAMES=YES, SAS variable names are generated from column names in the first row of the range. GETNAMES=YES also sets the SKIPROWS value to 1.

You can change the default value from NO to YES by setting the SS_NAMES environment variable. See “Setting Environment Variables” on page 142 for more information on setting and changing environment variables.

The GETNAMES statement is a database-description statement. It must follow the CREATE statement and precede any editing statements when you create a descriptor.

MIXED <=> YES | NO | Y | N;
determines whether to convert Microsoft Excel numeric data values in a column to their character representation when the corresponding SAS variable is expecting a character value.

The MIXED statement is optional. Use the MIXED statement if you have both Microsoft Excel numeric and character data in a column. Specifying YES allows both numeric and character data to be displayed as SAS character data. NO, the default, treats any data in a column that does not match the specified data type as missing values.

You can change the default value to YES by setting the SS_MIXED environment variable. See “Setting Environment Variables” on page 142 for more information on setting and changing environment variables.

The MIXED statement is an editing statement, and it must follow any database descriptions when you create an access descriptor.

RANGE <=> <'>*range-name*<'> | '*range-address*';
subsets a specified section of an XLS file worksheet. The *range-name* is the name that is assigned to a range address within the worksheet. Range names can be up to 15 characters long and are not case-sensitive.

The *range-address* is identified by the top left cell that begins the range and the bottom right cell that ends the range within the XLS worksheet file. The

beginning and ending cells are separated by two periods; for example, the range address C9..F12 indicates a cell range that begins at cell C9, ends at cell F12 and includes all cells in between.

The RANGE statement is optional. If you omit RANGE, the entire worksheet is accessed as the default range.

The RANGE is a database-description statement. It must follow the CREATE statement and precede any editing statements when you create a descriptor.

SCANTYPE <=> YES | NO | Y | N | <*number-of-rows*>;

finds the most common Excel data type and format for each column in a specified number of rows in an XLS worksheet to generate the SAS format. By default, SAS variable formats are generated from the Excel formats found in the first row of the entire worksheet or in the first row of a range (if specified) in the worksheet.

The SCANTYPE statement is optional, and its default value is NO. If you specify YES, the ACCESS procedure scans the data types and formats of all rows in each column of the worksheet or range and uses the most common one to generate the default SAS format for each column. If you specify a number of rows, PROC ACCESS scans the specified number of rows only and returns the most common format.

If you specify the SKIPROWS statement, the ACCESS procedure skips the specified rows and starts scanning from the next row. For example, if you specify SKIPROWS=3, PROC ACCESS skips the first three rows and begins scanning the data type and format on the fourth row.

You can change the default value to YES by setting the SS_SCAN environment variable. See “Setting Environment Variables” on page 142 for more information on setting and changing environment variables.

Specifying SCANTYPE=0 is equivalent to specifying SCANTYPE=NO.

The SCANTYPE statement is a database-description statement. It must follow the CREATE statement and precede any editing statements when you create a descriptor.

SKIPROWS <=> *number-of-rows-to-skip*;

specifies the number rows, beginning at the top of the range in the XLS file, to ignore when reading data from the XLS file. The default value for SKIPROWS is 0. The skipped (or ignored) rows often contain information such as column labels or names or underscores rather than input data.

If GETNAMES=YES, the default value of SKIPROWS automatically changes to 1. The first row of data and formats after SKIPROWS in a range is used to generate the SAS variable types and formats. However, you can use the SCANTYPE statement to scan the formats of a specified number of rows and to use the most common data type and format to generate the default SAS variable types and formats. See “Setting Environment Variables” on page 142 for more information on setting and changing environment variables.

The SKIPROWS statement is a database-description statement. It must follow the CREATE statement and precede any editing statements when you create a descriptor.

TYPE *column-identifier-1* <=> C | N < . . . *column-identifier-n* <=> C | N >;

changes the expected data types of SAS variables. SAS data sets have two data types: character (C) and numeric (N). Spreadsheet files have the same two data types: character (for labels and formula strings) and numeric (for numbers and formulas). Changing the default data type of a SAS variable in a descriptor file also changes its associated default format in the loaded file.

If you omit the TYPE statement, the database field types are generated from the PC file data types. You can change as many database field types as you want in one TYPE statement.

WORKSHEET <=> <'>*worksheet-name*<'>;
identifies one worksheet from a group of worksheets while you are reading from an XLS file. The *worksheet-name* is a 31-character name and is not case-sensitive. For example, specifying WORKSHEET=SHEET2 identifies worksheet 2 from a group of worksheets

The WORKSHEET statement is optional. For Excel 4 files, there is only one worksheet identifier, WORKSHEET1; therefore, the WORKSHEET statement is ignored. Under Excel 5, the default value is SHEET1. If you change the default worksheet from within Excel, you can either supply the new worksheet name or supply the worksheet's value (such as **`Sheet5`**).

The WORKSHEET statement is a database-description statement. It must follow the CREATE statement and precede any editing statements when you create an access descriptor.

DBLOAD Procedure: XLS Specifics

Chapter 4, "DBLOAD Procedure Reference," on page 41 describes the generic options and procedure statements that enable you to create a PC file format table and to insert data in it. The following section describes the file-specific statements you use in the SAS/ACCESS interface to XLS.

DBLOAD Procedure Statements for XLS

To create and load an XLS table, the SAS/ACCESS interface to XLS uses the following statements in batch mode:

PROC DBLOAD DBMS=XLS | EXCEL <DATA=<*libref.*>*SAS-data-set*>;
PATH='*path-and-filename*<.XLS>' | <'>*filename*<'> | *fileref*;
VERSION <=> *EXCEL-product-number*;
PUTNAMES <=> YES | NO | Y | N;
ACCDESC= <*libref.*>*access-descriptor*;
DELETE *variable-identifier-1* <*...variable-identifier-n*>;
ERRLIMIT= *error-limit*;
FORMAT *SAS-variable-name-1 SAS-format-1* <=>
<*...SAS-variable-name-n SAS-format-n*>;
LABEL;
LIMIT=*load-limit*;
LIST <ALL | COLUMNS | FIELDS | *variable-identifier*>;
RENAME *variable-identifier-1* <=> <'>*column-name-1*<'>
<*...variable-identifier-n* = <'>*column-name-n*<'>>;
RESET ALL | *variable-identifier-1* <*...variable-identifier-n*>;
WHERE *SAS-where-expression* ;
LOAD ;
RUN ;

The QUIT statement is also available in PROC DBLOAD. However, its use causes the procedure to terminate. QUIT is used most often in the interactive line and noninteractive modes to exit the procedure without exiting SAS.

FORMAT *SAS-variable-name-1 SAS-format-1 <...SAS-variable-name-n SAS-format-n>*;
assigns a temporary format to a SAS variable in the input SAS data set. This format temporarily overrides any other format for the variable. The assignment lasts only for the duration of the procedure. Assign formats to as many variables as you want in one FORMAT statement.

Use FORMAT when you want to change the format, column width, or the number of decimal digits for columns being loaded into the PC file. For example, if you change the SAS variable format 12.1 to DOLLAR15.2, the column format of the loaded data changes from a fixed numeric format with a column width of 12 and one decimal digit to a currency format with a column width of 15 and two decimal digits.

PUTNAMES <=> YES|NO|Y|N;
writes column names to the first row of the new XLS file. The column names can be default SAS variables names or, if you specify the LABEL statement, SAS variable labels. You can modify the column names using the RENAME statement.

The PUTNAMES statement is optional. If you omit PUTNAMES, data are read from the data set and written to the XLS file beginning in the first row of the XLS file, and no column names are written to the file.

You can change the default value to YES by setting the SS_NAMES environment variable. See "Setting Environment Variables" on page 142 for more information on setting and changing environment variables.

VERSION <=> *EXCEL-product-number*;
specifies the version number of the Excel product you are using, such as Excel 5.0. The *EXCEL-product-number* argument can be one of the following values: 3, 4, 5, or 7.

The DBLOAD procedure chooses the default version of Excel depending on which operating environment you use. If you use Windows, DBLOAD uses Excel 5.0. Excel 5 files have the identical format to Excel 95 (Version 7) files. If you use OS/2, DBLOAD uses Excel 4.0 because OS/2 does not support OLE2.

PROC DBLOAD does not support Excel 97 (Version 8) files. For information about accessing these files, see "Understanding XLS Essentials" on page 132.

Specify VERSION before the TYPE statement in order to get the correct data types for your new .XLS table.

How the SAS/ACCESS Interface Works

The SAS/ACCESS interface accesses data in the Microsoft Excel XLS files directly. It enables you to create SAS data sets from XLS files or directly read the XLS file data without creating SAS data sets. The interface does not allow you to update, add, or delete data in XLS files.

Accessing the Data

To access the data, the interface accesses a range in a worksheet as a table. If the range is not specified, the interface accesses the entire worksheet as a table. By default, the interface uses the Microsoft Excel formats of columns in the first row of the range to determine the formats of variables in SAS/ACCESS descriptors.

However, you can manipulate where the interface begins to read data and what format the interface generates by using the SKIPROWS and SCANTYPE statements in the ACCESS procedure. SKIPROWS skips a specified number of rows before reading data. SCANTYPE finds the most common data type and format from among a specified number of rows within an XLS range (after skipping the number of rows specified in SKIPROWS) and uses it to generate the default data type and format for SAS variables.

The ACCESS procedure enables you to create access descriptors and view descriptors for XLS files. You then can use the view descriptors as SAS data sets.

You can retrieve a subset of data using the WHERE statement .

To sort XLS file data, you must first extract the data from an XLS file and place them in a SAS data file, unless you are using the SQL procedure. (The SQL procedure enables you to present output data in a sorted order using the ORDER BY clause of the SELECT statement.) You can extract and sort XLS file data in one step with the OUT= option in the SORT procedure, using a view to the XLS file as input to PROC SORT.

Creating and Loading the Data

When you use PROC DBLOAD to create and load XLS files, the procedure translates the SAS data set into an XLS file. The file is stored in the location specified by the PATH= statement. Only one SAS data set can be loaded into an XLS file at one time. The loaded XLS file can contain only one worksheet. Microsoft Excel then reads data from the loaded XLS file directly.

In the DBLOAD procedure, you can specify the PUTNAMES statement to place the SAS variable names in the first row of the spreadsheet and the first observation in the second row, and so on. If PUTNAMES is not specified, the first observation is placed in the first row, the second observation is placed in the second row, and so on. Columns do not have names. The formats for SAS variables are automatically converted to the closest corresponding Microsoft Excel data types and formats. See the descriptions of individual statements for more information on how the data and columns are read.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/ACCESS® Software for PC File Formats: Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS/ACCESS® Software for PC File Formats: Reference, Verison 8

ISBN 1-58025–544–2

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999