



CHAPTER

4

Using External Files

<i>About External Files</i>	82
<i>Referencing External Files</i>	82
<i>Using a Fileref</i>	83
<i>Assigning a File Shortcut</i>	83
<i>Using the FILENAME Statement</i>	83
<i>Using Environment Variables</i>	84
<i>Using the SET system option</i>	85
<i>Using the SET command</i>	85
<i>Assigning a Fileref to a Directory</i>	85
<i>Assigning a Fileref to Concatenated Directories</i>	87
<i>Summary of Rules for Resolving Member Name Syntax</i>	87
<i>Assigning a Fileref to Concatenated Files</i>	88
<i>Referencing External Files with Long Filenames</i>	88
<i>Listing Fileref Assignments</i>	88
<i>Clearing Filerefs</i>	89
<i>Understanding How Concatenated Directories Are Accessed</i>	89
<i>Input and update</i>	89
<i>Output</i>	90
<i>Understanding How Concatenated Files Are Accessed</i>	90
<i>Input</i>	90
<i>Output and update</i>	90
<i>Using a Quoted OS/2 Filename</i>	90
<i>Using Reserved Operating Environment Physical Names</i>	91
<i>Using a File in Your Working Directory</i>	92
<i>Accessing External Files with SAS Statements</i>	92
<i>Using the FILE Statement</i>	92
<i>Using the INFILE Statement</i>	93
<i>Using the %INCLUDE Statement</i>	94
<i>Accessing External Files with SAS Commands</i>	95
<i>Using the FILE Command</i>	95
<i>Using the INCLUDE Command</i>	95
<i>Using the GSUBMIT Command</i>	96
<i>Advanced External I/O Techniques</i>	96
<i>Altering the Record Format</i>	96
<i>Appending Data to an External File</i>	97
<i>Determining Your Hard Drive Mapping</i>	97
<i>Reading External Files with National Characters</i>	98
<i>Reading Data from the Communications Port</i>	98
<i>Communications Port Timeouts</i>	99
<i>Options that Relate to Communications Port Timeouts</i>	100
<i>Reading Data Using DataMyte Processing</i>	100

About External Files

External files are files that contain data or text, such as SAS programming statements, records of raw data, or procedure output. The SAS System can use these files, but they are not managed by the SAS System.

SAS Language Reference: Concepts contains basic, platform-independent information about external files.

For information about how to access external files that contain transport data libraries, see the SAS Technical Support Web page.

Referencing External Files

To access external files, you must tell the SAS System how to find the files. Use the following statements to access external files:

FILENAME

associates a fileref with an external file that is used for input or output.

FILE

opens an external file for writing data lines. Use the PUT statement to write lines.

INFILE

opens an external file for reading data lines. Use the INPUT statement to read lines.

%INCLUDE

opens an external file and reads SAS statements from that file. (No other statements are necessary.)

These statements are discussed later in this section and in “SAS Statements under OS/2” on page 289, as well as in the SAS language statements section in *SAS Language Reference: Dictionary*.

You can also specify external files in various SAS dialog entry fields (for example, as a file destination in the Save As dialog box), in the FILENAME function, and in SAS commands, such as FILE and INCLUDE.

Depending on the context, SAS can reference an external file by using

- a fileref
- an environment variable that is defined with either the SET system option or the OS/2 SET command
- an OS/2 filename that is enclosed in quotes
- member name syntax (also called aggregate syntax)
- a single filename without quotes (a file in the working directory).

The following sections discuss these methods of specifying external files.

Because there are several ways to specify external files in the SAS System, SAS uses a set of rules to resolve an external file reference. The following list represents the order of precedence:

- 1 Check for a standard OS/2 file specification that is enclosed in quotes.
- 2 Check for a fileref that is defined by a FILENAME statement or function.
- 3 Check for an environment variable fileref.
- 4 Assume that the file is in the working directory.

In other words, the SAS System assumes that an external file reference is a standard OS/2 file specification. If it is not, SAS checks to see if the file reference is a fileref

(defined by either a FILENAME statement, FILENAME function, or an environment variable). If the file reference is none of these, SAS assumes that it is a filename in the working directory. If the external file reference is not valid for one of these choices, SAS issues an error message that indicates that it cannot access the external file.

Using a Fileref

One way to reference external files is with a *fileref*. A fileref is a logical name that is associated with an external file. You can assign a fileref with a File Shortcut in the SAS Explorer window, the My Favorite Folders window, the FILENAME statement, and the FILENAME function, or you can use an OS/2 environment variable to point to the file. This section discusses the different ways to assign filerefs. It also shows you how to obtain a listing of the active filerefs and clear filerefs during your SAS session.

Assigning a File Shortcut

In an interactive SAS session, you can use the SAS Explorer window or the My Favorite Folders window to create filerefs. The SAS Explorer File Shortcuts folder contains a list of active filerefs. To create a new fileref from SAS Explorer:

- 1 Select the File Shortcuts folder and then select



- 2 In the New... dialog box, select **File Shortcuts** and click on **OK**.
- 3 In the New File Shortcut window, type the name of the shortcut (fileref) and the path to the SAS file that the shortcut represents.

To assign a file shortcut by using the My Favorite Folders window:

- 1 Open the folder that contains the file.
- 2 Position the cursor over the file, click with the right mouse button and select **Create File Shortcut**.
- 3 In the Create File Shortcut dialog box, type the name of the file shortcut and click on **OK**.

You can then use these file shortcuts in your SAS programs.

Note: File shortcuts are active only during the current SAS session. △

Using the FILENAME Statement

The FILENAME statement enables you to associate a logical name with an external file, directory, or device.

Note: The syntax of the FILENAME function is similar to the FILENAME statement. For information about the FILENAME function, see *SAS Language Reference: Dictionary*. △

The simplest form of the FILENAME statement is as follows:

```
FILENAME fileref "external-file";
```

For example, if you want to read the file C:\MYDATA\SCORES.DAT, you can issue the following statement to associate the fileref MYDATA with the file C:\MYDATA\SCORES.DAT:

```
filename mydata "c:\mydata\scores.dat";
```

Then you can use this fileref in your SAS programs. For example, the following statements create a SAS data set that is named TEST, by using the data that are stored in the external file that is referenced by the fileref MYDATA:

```
data test;
  infile mydata;
  input name $ score;
run;
```

CAUTION:

Under OS/2, the words CON, NUL, KBDS, PRN, COM1 - COM3, and LPT1 - LPT3 are reserved words. Do not use these reserved words as a fileref, libref, variable name, or macro variable. Δ

You can also use the FILENAME statement to concatenate directories of external files and to concatenate multiple individual external files into one logical external file. These topics are discussed in “Assigning a Fileref to Concatenated Directories” on page 87 and “Assigning a Fileref to Concatenated Files” on page 88.

The asterisk (*) and question mark (?) wildcards can be used in either the external filename or in the file extension to match input file names. Use * to match one or more characters and the ? to match a single character. Wildcards are supported for input only in the FILENAME and INFILE statements, and in member name syntax (aggregate syntax). Wildcards are not valid in the FILE statement. The following filename statement reads input from every file in the current directory that begins with the string **wild** and ends with **.dat**:

```
filename wild 'wild*.dat';
data;
  infile wild;
  input;
run;
```

The following example reads all files in the current working directory:

```
filename allfiles '*.*';
data;
  infile allfiles;
  input;
run;
```

The FILENAME statement accepts various options that enable you to associate device names, such as printers, with external files and to control file characteristics, such as record format and length. Some of these options are shown in “Advanced External I/O Techniques” on page 96. For the complete syntax of the FILENAME statement, see to “FILENAME” on page 292.

Using Environment Variables

Just as you can define an environment variable to serve as a logical name for a SAS data library (see “Assigning SAS Libraries Using Environment Variables” on page 65), you can also use an environment variable to refer to an external file. You can choose either to define a SAS environment variable by using the SET system option or to define an OS/2 environment variable by using the OS/2 SET command.

CAUTION:

Under OS/2, the words CON, NUL, KBDS, PRN, COM1 - COM3, and LPT1 - LPT3 are reserved words. Do not use these reserved words as an environment variable. Δ

The availability of environment variables makes it simple to assign resources to the SAS System prior to invocation. However, the environment variables that you define

(using the SET system option) for a particular SAS session are not available to other applications.

Using the SET system option

For example, to define a SAS environment variable that points to the external file C:\MYDATA\TEST.DAT, you can use the following SET option in your SAS configuration file:

```
-set myvar c:\mydata\test.dat
```

Then, in your SAS programs, you can use the environment variable MYVAR to refer to the external file:

```
data mytest;
  infile myvar;
  input name $ score;
run;
```

It is recommended that you use the SET system option in your SAS configuration file if you invoke the SAS System through a program group window.

Using the SET command

An alternative to using the SET system option to define an environment variable is to use the OS/2 SET command. For example, the OS/2 SET command that is equivalent to the previous example is

```
SET MYVAR=C:\MYDATA\TEST.BAT
```

You must issue all the SET commands that define your environment variables before you invoke the SAS System.

When you reference an external environment variable (one that is assigned with the OS/2 SET command) in your SAS programs (such as in a FILE statement), a note that informs you that the environment variable has been used and that the fileref has been assigned is written to the SAS log.

Assigning a Fileref to a Directory

You can assign a fileref to a directory and then access individual files within that directory using member name syntax (also called aggregate syntax).

For example, if all your regional sales data for January are stored in the directory C:\SAS\MYDATA, you can issue the following FILENAME statement to assign the fileref JAN to this directory:

```
filename jan "c:\sas\mydata";
```

Now you can use this fileref with a member name in your SAS programs. In the following example, you reference two files that are stored in the JAN directory:

```
data westsale;
  infile jan(west);
  input name $ 1-16 sales 18-25
         comiss 27-34;
run;
data eastsale;
  infile jan(east);
  input name $ 1-16 sales 18-25
         comiss 27-34;
```

```
run;
```

When you use member name syntax, you do not have to specify the file extension for the file that you are referencing, as long as the file extension is the expected one. For instance, in the previous example, the INFILE statement expects a file extension of .DAT. Table 4.1 on page 86 lists the expected file extensions for the various SAS statements and commands, and lists the window in which the statement or command is used.

Table 4.1 Default File Extensions for Referencing External Files with Member Name Syntax

SAS Command or Statement	SAS Window	File Extension
FILE statement	Program	.DAT
	Editor	
%INCLUDE statement	Program	.SAS
	Editor	
INFILE statement	Program	.DAT
	Editor	
FILE command	Program	.SAS
	Editor	
FILE command	Log	.LOG
FILE command	Output	.LST
FILE command	NOTEPAD	none
INCLUDE command	Program	.SAS
	Editor	
INCLUDE command	NOTEPAD	none

For example, the following program submits the file C:\PROGRAMS\TESTPGM.SAS to the SAS System:

```
filename test "c:\programs";
%include test(testpgm);
```

The SAS System searches for a file that is named TESTPGM.SAS in the directory C:\PROGRAMS.

If your file has a file extension that is different from the default file extension, you can use the file extension in the filename, as in the following example:

```
filename test "c:\programs";
%include test(testpgm.xyz);
```

If your file has no file extension, you must enclose the filename in quotes, as in the following example:

```
filename test "c:\programs";
%include test("testpgm");
```

For more illustrations of the default file extensions that the SAS System uses, see the following examples. Assume that the following FILENAME statement has been submitted:

```
filename test "c:\mysasdir";
```

The following example opens the file C:\MYSASDIR\PGM1.DAT for output:

```
file test(pgml);
```

The following example opens the file C:\MYSASDIR\PGM1.DAT for input:

```
infile test(pgml);
```

The following example reads and submits the file C:\MYSASDIR\PGM1:

```
%include test("pgml");
```

These examples use SAS statements. SAS commands, such as the FILE and INCLUDE commands, also accept member name syntax and have the same default file extensions as shown in Table 4.1 on page 86.

Another feature of member name syntax is that it enables you to reference a subdirectory in the working directory without using a fileref. As an example, suppose you have a subdirectory named PROGRAMS that is located in a subdirectory of the working directory. You can use the subdirectory name PROGRAMS when referencing files within this directory. For example, the following statement submits the program that is stored in *working-directory*\PROGRAMS\PGM1.SAS:

```
%include programs(pgml);
```

The next example uses the FILE command to save the contents of the active window to *working-directory*\PROGRAMS\TESTPGM.DAT:

```
file programs(testpgm);
```

Note: If a directory name is the same as a previously defined fileref, the fileref takes precedence over the directory name. △

Assigning a Fileref to Concatenated Directories

Member name syntax is also handy when you use the FILENAME statement to concatenate directories of external files. For example, suppose you issue the following FILENAME statement:

```
filename progs ("c:\sas\programs",
               "d:\myprogs");
```

This statement tells the SAS System that the fileref PROGS refers to all files that are stored in both the C:\SAS\PROGRAMS and the D:\MYPROGS directories. When you use the fileref PROGS in your SAS program, the SAS System looks in these directories for the member that you specify. When you use this concatenation feature, you should be aware of the protocol that the SAS System uses, which depends on whether you are accessing the files to read, write, or update them. For more information, see “Understanding How Concatenated Directories Are Accessed” on page 89.

Summary of Rules for Resolving Member Name Syntax

The SAS System resolves an external file reference that uses member name syntax by using a set of rules. For example, suppose your external file reference in a SAS statement or command is the following:

```
progs(member1)
```

The SAS System uses the following set of rules to resolve this external file reference. This list represents the order of precedence:

- 1 Check for a fileref named PROGS that is defined by a FILENAME statement.

- 2 Check for a SAS or OS/2 environment variable that is named PROGS.
- 3 Check for a directory that is named PROGS beneath the working directory.

The member name must be a valid physical filename. If no extension is given (as in the previous example), the SAS System uses the appropriate default extension, as given in Table 4.1 on page 86. If the extension is given or if the member name is quoted, the SAS System does not assign an extension, and it looks for the filename exactly as it is given.

Assigning a Fileref to Concatenated Files

You can specify concatenations of files when you are reading external files from within the SAS System. Concatenated files consist of two or more file specifications (which may contain wildcard characters) that are separated by blanks or commas. The following are some examples of valid concatenation specifications:

- `filename allsas ("one.sas", "two.sas", "three.sas");`
- `filename alldata ("test1.dat" "test2.dat" "test3.dat");`
- `filename allinc "test*.sas";`
- `%include allsas;`
- `infile alldata;`
- `include allinc`

Note: The FSLIST procedure in FSP does not support concatenated files. Δ

When you use this concatenation feature, note the protocol that the SAS System uses; the protocol depends on whether you are accessing the files in order to read, write, or update them. For more information, see “Understanding How Concatenated Files Are Accessed” on page 90.

Note: Do not confuse concatenated file specifications with concatenated *directory* specifications, which are also valid and are shown in “Assigning a Fileref to Concatenated Directories” on page 87. Δ

Referencing External Files with Long Filenames

The SAS System under OS/2 supports the use of long filenames on disk volumes that are formatted with the high-performance file system (HPFS). You can use long filenames whenever you specify a command, or an element of the SAS language, or a filename as an argument to a dialog box. For more information about valid long filenames, see *OS/2 Environment: Changes and Enhancements to the SAS System* and your OS/2 operating environment documentation.

When you are specifying external filenames by using the SAS language, such as specifying the filename in a statement or function, enclose the filename in double quotes to reduce ambiguity (a single quote is a valid character in a long filename). When you need to specify multiple filenames, enclose each filename in double quotes and delimit the names with a blank space.

Here are some examples of valid uses of long filenames within SAS:

- `libname abc "My data file";`
- `filename myfile "Bernie's file";`
- `filename summer ("June sales" "July sales" "August sales");`
- `include "A really, really big SAS program";`

Listing Fileref Assignments

If you have assigned several filerefs during a SAS session and need to remember which fileref points to which file, you can use either the SAS Explorer window or the FILENAME statement to list all the assigned filerefs.

To use the SAS Explorer window to list the active filerefs, double-click on the **File Shortcuts** icon. The Explorer window lists all the filerefs that are active for your current SAS session. Any environment variables that you have defined as filerefs are listed, provided that you have used them in your SAS session. If you have defined an environment variable as a fileref but have not yet used it in a SAS program, the fileref is not listed in the Explorer window.

If you are running the SAS System in batch mode, you can use the following FILENAME statement to write the active filerefs to the SAS log:

```
filename _all_ list;
```

Clearing Filerefs

You can clear a fileref by submitting the FILENAME statement and by using the following syntax:

```
FILENAME fileref | _ALL_ <CLEAR>;
```

If you specify a fileref, only that fileref is cleared. If you specify the keyword `_ALL_`, all the filerefs that you have assigned during your current SAS session are cleared.

To clear filerefs by using SAS Explorer:

- 1 double-click on **File Shortcuts**
- 2 select the file shortcuts that you want to delete. To select all file shortcuts, select

►

- 3 press the DELETE key or select

►

- 4 Click on in the message box to confirm deletion of the file shortcuts.

Note: When you clear a fileref that is defined by an environment variable, the variable remains defined, but it is no longer considered a fileref. That is, it is no longer listed in the SAS Explorer window. You can use the variable in another FILENAME statement to create a new fileref. △

The SAS System automatically clears the association between filerefs and their respective files at the end of your job or session. If you want to associate the fileref with a different file during the current session, you do not have to end the session or clear the fileref. The SAS System automatically reassigns the fileref when you issue a FILENAME statement for the new file.

Understanding How Concatenated Directories Are Accessed

When you associate a fileref that has more than one physical directory, the file that is accessed depends upon whether you want access to it for input, output, or update.

Input and update

If the file is opened for input or update, the first file that is found that matches the member name is accessed. For example, if you submit the following statements, and the file PHONE.DAT exists in both the C:\SAMPLES and C:\TESTPGMS directories, the one in C:\SAMPLES is read:

```
filename test ("c:\samples","c:\testpgms");
data sampdat;
  infile test(phone.dat);
  input name $ phonenum $ city $ state $;
```

```
run;
```

Output

When you open a file for output, the SAS System writes to the file in the first directory that is listed in the FILENAME statement, even if a file by the same name exists in a later directory. For example, suppose that you execute the following FILENAME statement:

```
filename test ("c:\sas","d:\mysasdir");
```

Then, if you issue the following FILE command, the file SOURCE.PGM is written to the C:\SAS directory, even if a file by the same name exists in the D:\MYSASDIR directory:

```
file test(source.pgm)
```

Understanding How Concatenated Files Are Accessed

When you associate a fileref that has more than one physical file, the behavior of SAS statements and commands depends on whether you are accessing the files for input, output, or update.

Note: You should not use concatenated files with some procedures, such as the FSLIST procedure. Δ

Input

If the file is opened for input, data from all files are input. For example, if you issue the following statements, the %INCLUDE statement submits four programs for execution:

```
filename mydata ("qtr1.sas","qtr2.sas",
                "qtr3.sas","qtr4.sas");
%include mydata;
```

Output and update

If the file is opened for output or update, data are written to the first file in the concatenation. For example, if you issue the following statements, the PUT statement writes to MYDAT1.DAT:

```
filename indata "dogdat.dat";
filename outdata ("mydat1.dat","mydat2.dat",
                 "mydat3.dat","mydat4.dat");
data _null_;
  infile indata;
  input name breed color;
  file outdata;
  put name= breed= color=;
run;
```

Using a Quoted OS/2 Filename

Instead of using a fileref to refer to external files, you can use a quoted OS/2 filename. For example, if the file C:\MYDIR\ORANGES.SAS contains a SAS program that you want to run, you can issue the following statement:

```
%include "c:\mydir\oranges.sas";
```

When you use a quoted OS/2 filename in a SAS statement, you can omit the drive and directory specifications if the file that you want to reference is located in the working directory. For instance, if in the previous example the working directory is C:\MYDIR, you can submit the following statement:

```
%include "oranges.sas";
```

Using Reserved Operating Environment Physical Names

You can use several reserved names as quoted physical filenames. Reserved operating environment physical names enable you to do a variety of things, such as read data directly from the communications port (such as COM1). Table 4.2 on page 91 lists these physical names and their corresponding device-type keywords:

Table 4.2 Reserved OS/2 Physical Names

Physical Name	Device Type	Use
COM1-COM4	COMMPORT	Read or write from the communications port.
NUL	DUMMY	Discard data. This name is useful in testing situations.

You can specify operating environment physical names with or without a colon. For example, you can specify either COM1: or COM1. For additional information about physical names, see your OS/2 documentation.

The following example demonstrates how to capture data from an external device or application that is transmitting data via a serial (RS-232C port).

```
options noxwait ssync;
data _null_;
  if symget("sysscpl") = "OS2" then
    rc = system("mode COM1:9600,n,8,1,xon=on");
  stop;
run;

filename commdata commport "COM1:";

data fruit;
  keep num type;
  infile commdata unbuffered;
  file commdata;
  put "ready";
  input totrecs records $;
  if totrecs = . or records ne "RECORDS" then
    do;
      file log;
      put "ERROR: Unable to determine
        number of records to read.";
    stop;
  end;
  do i = 1 to totrecs;
    input num type $;
    output;
```

```

        put "NEXT";
    end;
    stop
run;
```

Note the use of the device-type keyword `COMMPORT` in the `FILENAME` statement in this example. Because the access protocols for devices are slightly different from the access protocols for files, always use the appropriate device-type keyword in combination with the reserved physical name in the `FILENAME` statement. If you do not use a device-type keyword, the SAS System defaults to using the access protocols for files, not for devices.

As an alternative to using a device-type keyword in the `FILENAME` statement, you can use the `DEVICE=` option in the `%INCLUDE` statement. Although this technique serves the same purpose as the device-type keyword, it is not as flexible an approach as using a device-type keyword in the `FILENAME` statement. For example, if you use a fileref in a `PROC PRINTTO` statement, no `DEVICE=` option is available.

For more information about available device-type keywords in the `FILENAME` statement and for valid values for the `DEVICE=` option in the `%INCLUDE` statement, see the descriptions of these statements in “SAS Statements under OS/2” on page 289. For a discussion of the access protocols for using a communications port device, see “Reading Data from the Communications Port” on page 98.

Using a File in Your Working Directory

If you store the external files that you need to access in your working directory, and they have the expected file extensions (see Table 4.1 on page 86), you can simply refer to the filename, without quotes or file extensions, in a SAS statement. For example, if you have stored a file that is named `ORANGES.SAS` in your working directory and `ORANGES` is not defined as a fileref, you can submit the file with the following statement:

```
%include oranges;
```

Remember, though, that using this type of file reference requires that

- the file be stored in the working directory
- the file have the correct file extension
- the filename not also be defined as a fileref.

For more information about how to determine and change the SAS working directory, see “Setting the Current Folder” on page 6 and “Changing the SAS Current Folder” on page 42.

Accessing External Files with SAS Statements

This section presents examples of using the `FILE`, `INFILE`, and `%INCLUDE` statements to access external files. The following examples are simple. For more complex examples of using these statements under OS/2, see “Advanced External I/O Techniques” on page 96.

Using the `FILE` Statement

The `FILE` statement enables you to direct lines that are written by a `PUT` statement to an external file.*

The following is a simple example that uses the FILE statement. This example reads the data in the SAS data set MYLIB.TEST and writes only those scores that are greater than 95 to the external file C:\MYDIR\TEST.DAT:

```
filename test "c:\mydir\test.dat";
libname mylib "c:\mydata";
data _null_;
  set mylib.test;
  file test;
  if score ge 95 then
    put score;
run;
```

The previous example illustrates writing the value of only one variable of each observation to an external file. The following example uses the `_ALL_` option in the PUT statement to copy all variables in the current observation to the external file if the variable REGION contains the value `west`.

```
libname us "c:\mydata";
data west;
  set us.pop;
  file "c:\sas\pop.dat";
  where region="west";
  put _all_;
run;
```

This technique of writing out entire observations is particularly useful if you need to write variable values in a SAS data set to an external file so that you can use your data with another application that cannot read data in a SAS data set format.

Note: This example uses the `_ALL_` keyword in the PUT statement. This code generates *named output*, which means that the variable name, an equal sign (=), and the variable value are all written to the file. Consider this when you are reading the data later. For more information about named output, see the description of the PUT statement in *SAS Language Reference: Dictionary*. △

The FILE statement also accepts several options. These options enable you, among other things, to control the record format and length. Some of these options are illustrated in “Advanced External I/O Techniques” on page 96. For the complete syntax of the FILE statement, see “FILE” on page 291.

Note: The default record length that is used by the FILE statement is 256 characters. If the data that you are saving have records longer than this, you must use the FILENAME statement to define a fileref and either use the LRECL= option in the FILENAME statement to specify the correct logical record length or specify the LRECL= option in the FILE statement. For details about the LRECL= option, see LRECL= in “FILE” on page 291. △

Using the INFILE Statement

The INFILE statement is used to specify the source of data that are read by the INPUT statement in a SAS DATA step. The source can be either a text file. The INFILE statement is always used in conjunction with an INPUT statement, which defines the location, order, and type of data being read.

* You can also use the FILE statement to direct PUT statement output to the SAS log or to the same destination as procedure output. For more information, see *SAS Language Reference: Dictionary*.

The following is a simple example of the INFILE statement. This DATA step reads the specified data from the external file and creates a SAS data set that is named SURVEY:

```
filename mydata "c:\mysasdir\survey.dat";
data survey;
  infile mydata;
  input fruit $ taste looks;
run;
```

Of course, you can use a quoted OS/2 filename instead of a fileref:

```
data survey;
  infile "c:\mysasdir\survey.dat";
  input fruit $ taste looks;
run;
```

The INFILE statement also accepts other options. These options enable you, among other things, to control the record format and length. Some of these options are illustrated in “Advanced External I/O Techniques” on page 96. For the complete syntax of the INFILE statement, see “INFILE” on page 299.

Note: The default record length that is used by the INFILE statement is 256 characters. If the data that you are reading have records that are longer than this, you must use the FILENAME statement to define a fileref and either use the LRECL= option in the FILENAME statement to specify the correct logical record length or specify the LRECL= option in the INFILE statement. For details about the LRECL= option, see LRECL= in “INFILE” on page 299. Δ

Using the %INCLUDE Statement

When you submit a %INCLUDE statement, it reads an entire file into the current SAS program that you are running and submits that file to the SAS System immediately. A single SAS program can have as many individual %INCLUDE statements as necessary, and you can nest up to ten levels of %INCLUDE statements. Using the %INCLUDE statement makes it easier for you to write modular SAS programs.

Here is an example that submits the statements that are stored in C:\SAS\MYJOBS\PROGRAM1.SAS by using the %INCLUDE statement and member name syntax:

```
filename job "c:\sas\myjobs";
%include job(program1);
```

The %INCLUDE statement also accepts several options. These options enable you, among other things, to control the record format and length. Some of these options are illustrated in “Advanced External I/O Techniques” on page 96. For the complete syntax of the %INCLUDE statement, see “Using the %INCLUDE Statement” on page 94.

Note: The default record length that is used by the %INCLUDE statement is 256 characters. If the program that you are reading has records longer than this, you must use the FILENAME statement to define a fileref and either use the LRECL= option in the FILENAME statement to specify the correct logical record length or specify the LRECL= option in the %INCLUDE statement. For details about the LRECL= option, see LRECL= in “FILE” on page 291. Δ

Accessing External Files with SAS Commands

This section illustrates how to use the FILE and INCLUDE commands to access external files. Commands provide the same purpose as the Save As and Open dialog boxes that are discussed in “Opening and Saving Files” on page 29. The method that you use to access external files depends on the needs of your SAS application and on your personal preference.

Using the FILE Command

The FILE command has a different use from the FILE statement; the FILE command writes the current contents of a window to an external file rather than merely specifying a destination for PUT statement output in a DATA step.

For example, if you want to save the contents of the Log window to an external file that is named C:\SASLOGS\TODAY.LOG, you can issue the following FILE command from the Command dialog box; however, the LOG window must be active.

```
file "c:\saslogs\today.log"
```

If you have already defined the fileref LOGS to point to the SASLOGS directory, you can use the following FILE command:

```
file logs(today)
```

In this case, the file extension defaults to .LOG, as shown in Table 4.1 on page 86.

If you use the FILE command to attempt to write to an already existing file, a dialog box gives you the choice of replacing the existing file, appending the contents of the window to the existing file, or canceling your request.

If you issue the FILE command with no arguments, the contents of the window are written to the file that is referenced in the last FILE command. This is useful if you are editing a program and want to save it often. However, the dialog box that prompts you about replacing or appending appears only the first time that you issue the FILE command with no arguments. Thereafter, unless you specify the filename in the FILE command, it uses the parameters that you specified earlier (replace or append) without prompting you.

Selecting the **Save As** item from the main SAS window **File** menu displays the Save As dialog box. This dialog box performs the same function as the FILE command, but it is more flexible in that it gives you more choices and is more interactive than the FILE command. For more information, see “Opening and Saving Files” on page 29.

The FILE command also accepts several options. These options enable you to associate a device with the output file and to control the record format and length. Some of these options are illustrated in “Advanced External I/O Techniques” on page 96. For the complete syntax of the FILE command, see “FILE” on page 210.

Using the INCLUDE Command

The INCLUDE command, like the %INCLUDE statement, can be used to copy an entire external file into the Program Editor window, the NOTEPAD window, or whatever window is active. In the case of the INCLUDE command, however, the file is simply copied to the window and is not submitted.

For example, suppose that you want to copy the file C:\SAS\PROG1.SAS into the PROGRAM EDITOR window. If you have defined a fileref SAMPLE to point to the correct directory, you can use the following INCLUDE command from the Command

dialog box, assuming the Program Editor is the active window, to copy the member PROG1 into the Program Editor window:

```
include sample(prog1)
```

Another way to copy files into your SAS session is to use the Open dialog box. In addition to copying files, the Open dialog box gives you other choices, such as running the program that you are copying. The Open dialog box is the most flexible way for you to copy files into the Program Editor window. For more information, see “Opening and Saving Files” on page 29.

The INCLUDE command also accepts several options. These options enable you to associate a device with the input file and to control the record format and length. Some of these options are illustrated in “Advanced External I/O Techniques” on page 96. For the complete syntax of the INCLUDE command, see “INCLUDE” on page 213.

Issuing the INCLUDE command with no arguments includes the file that is referenced in the last INCLUDE command. If no previous INCLUDE command exists, you receive an error message.

Using the GSUBMIT Command

The GSUBMIT command can be used to submit SAS statements that are stored in the OS/2 clipboard. To submit SAS statements from the clipboard, use the following command:

```
gsubmit buffer=default
```

You can also use the GSUBMIT command to submit SAS statements that are specified as part of the command. For more information about the GSUBMIT command, see *SAS Language Reference: Dictionary*.

Advanced External I/O Techniques

This section illustrates how to use the FILENAME, FILE, and INFILE statements to perform more advanced I/O tasks, such as altering the record format and length, appending data to a file, and using the DRIVEMAP device-type keyword to determine which hard drives are available.

Altering the Record Format

Using the RECFM= option in the FILENAME, FILE, %INCLUDE, and INFILE statements enables you to specify the record format of your external files. The following example shows you how to use this option.

Usually, the SAS System reads a line of data until a carriage return and linefeed character combination (the hexadecimal value is '0DOA' x) are encountered or until just a linefeed character (the hexadecimal value is '0A0' x) is encountered. However, sometimes data do not contain these carriage control characters but do have fixed-length records. In this case, you can specify RECFM=F to read your data.

To read such a file, you need to use the LRECL= option to specify the record length and the RECFM= option to tell the SAS System that the records have fixed-length record format. Here are the required statements:

```
data test;
  infile "test.dat" lrecl=60 recfm=f;
  input x y z;
```



```
run;
```

In this example, the SAS System expects fixed-length records that are 60 bytes long, and it reads in the three numeric variables: X, Y, and Z.

You can also specify RECFM=F when your data do contain carriage returns and linefeeds, but you want to read these values as part of your data instead of treating them as carriage control characters. When you specify RECFM=F, the SAS System ignores any carriage controls and linefeeds and simply reads the record length that you specify.

Appending Data to an External File

Occasionally, you may not want to create a new output file, but rather append data to the end of an existing file. In this case, you can use the MOD option in the FILE statement as in the following example:

```
filename myfile "c:\sas\data";
data _null_;
  infile myfile(newdata);
  input sales expenses;
  file myfile(jandata) mod;
  put sales expenses;
run;
```

This example reads the variables SALES and EXPENSES from the external data file C:\SAS\DATA\NEWDATA.DAT and appends records to the existing data file C:\SAS\DATA\JANDATA.DAT.

If you are going to append data to several files in a single directory, you can use the MOD option in the FILENAME statement instead of in the FILE statement. You can also use the FAPPEND function or the PRINTTO procedure to append data to a file. For more information, see the SAS functions section in *SAS Language Reference: Dictionary* and the section on the PRINTTO procedure in *SAS Procedures Guide*.

Determining Your Hard Drive Mapping

You can use the DRIVEMAP device-type keyword in the FILENAME statement to determine which hard drives are available for use. Here is an example that uses this keyword:

```
filename myfile drivemap;
data mymap;
  infile myfile;
  input drive $;
  put drive;
run;
```

The information that is written to the SAS log looks similar to that shown in Output 4.1 on page 97.

Output 4.1 Drive Mapping Information

```

50 filename myfile drivemap;
51
52 data mymap;
53   infile myfile;
54   input drive $;
55   put drive;
56 run;
NOTE: The infile MYFILE is:
      FILENAME=DRIVEMAP,
      RECFM=V,LRECL=256
A:
C:
H:
J:
K:
L:
M:
N:
R:
S:
T:
U:
NOTE: 12 records were read from the infile MYFILE.
      The minimum record length was 2.
      The maximum record length was 2.
NOTE: The data set WORK.MYMAP has 12 observations
      and 1 variables.
NOTE: The DATA statement used 2.04 seconds.

```

You can use this technique in SAS/AF applications, where you can build selection lists to let a user choose a hard drive. You can also use the DRIVEMAP keyword to enable you to assign macro variables to the various available hard drives.

If you try to use the fileref that is associated with the DRIVEMAP device-type keyword in a WRITE or UPDATE situation, you receive an error message that tells you that you do not have sufficient authority to write to the file.

Reading External Files with National Characters

Most OS/2 applications read and write character data by using OEM character codes. In Version 8, SAS does not allow you to read or write to files by using OEM character sets.

Characters such as the \hat{A} are considered national characters. OS/2 represents each character with a hexadecimal number. If your external file was created with an OS/2 editor or in the SAS System under OS/2, you do not need to do anything. Simply read the file by using the FILENAME or FILE statements, as you would normally do.

Reading Data from the Communications Port

You can read data directly from the communications (serial) port on your machine. To set the serial communications parameters, use the port configuration tools that are in the OS/2 Workplace Shell to set up the communications port. The communications parameters that you specify are specific to each data collection device.

After you invoke the SAS System, submit a FILENAME statement to associate a fileref with the communications port, as in the following example:

```
filename test commport "com1:";
```

This FILENAME statement defines the fileref TEST, uses the COMMPORT device-type keyword that specifies that you are going to use a communications port, and specifies the COM1: reserved physical name.

Next, read the data from COM1: into a SAS data set by using the TEST fileref. The following DATA step reads in the data, 1 byte at a time, until the SAS System encounters an end-of-file character (the hexadecimal value of the end-of-file character is '1a'x):

```
data acquire;
  infile test lrecl=1 recfm=f unbuffered;
  input i $;
  /* Read until you find an end-of-file. */
  if i='1a'x then stop;
run;
```

The communications port can be accessed multiple times. However, while multiple reads are allowed, only one user at a time can write to the port.

Two functions that are useful in data acquisition applications are SLEEP and WAKEUP. These functions enable you to control when your program begins to execute. For example, you can use the WAKEUP function to cause your program to begin executing at exactly 2:00 a.m. For more information on these two functions, see "SLEEP" on page 258 and "WAKEUP" on page 260.

Communications Port Timeouts

By default, if you are reading from a communications port and a timeout occurs, an end-of-file (EOF) character is returned to the program. You can specify how communications port timeouts are handled by using the COMTIMEOUT= option. The COMTIMEOUT= option is valid in the FILENAME statement and must be used in conjunction with the COMMPORT device-type keyword in the FILENAME statement.

The COMTIMEOUT= option accepts the following values:

EOF	returns an end-of-file character when a timeout occurs. This is the default behavior. This causes the current DATA step to terminate.
WAIT	instructs the communications port to wait forever for data. In other words, this value overrides the timeout. In this case, no record is returned to the DATA step until data are available. This can cause your program to go into an infinite loop, so use this value with caution.
ZERO	returns a record length of 0 bytes when a timeout occurs. However, the DATA step does not terminate; it simply tries to read data again.

Here is an example of a FILENAME statement that specifies that a record length of 0 bytes be returned to the program when a timeout occurs:

```
filename test commport "com1:"
  comtimeout=EOF;
data test;
  infile test length=linelen recfm=F;
  input @;
  if linelen ne 0 then input value;
  else put 'Timeout reading from COM1:.';
run;
```

Options that Relate to Communications Port Timeouts

These options relate to the communications port timeouts.

RTIMEOUT	sets the wait on a read timeout in units of 0.01 second. A value of 100 means 1 second.
WTIMEOUT	sets the write timeout in units of 0.01 second. A value of 100 means 1 second; a value of 0 means it waits forever.

Reading Data Using DataMyte Processing

The SAS System under Windows supports DataMyte data collection devices through three SAS functions and one CALL routine. These functions are the DMYTECHC, DMYTECWD, and CMYTERVC functions. The CALL routine is DMYTECKS. These functions and the CALL routine are described in “SAS Functions under OS/2” on page 245 and “SAS CALL Routines under OS/2” on page 245.

A full discussion of DataMyte processing is beyond the scope of this book; this section covers only the main points. A DataMyte is a data collection device that you attach to your communications port. The DataMyte device is typically used to interface with precision instruments in industrial and factory applications.

You can send data to and request data from the DataMyte device. A chunk of data that are passed at one time is called a packet. Each packet can be up to 255 characters long and can consist of several components. Two of these components are used by the SAS functions that support DataMyte data collection:

- character count (CC), which is the number of characters in the packet excluding the start-of-text, the character count itself, and the checksum.
- checksum (CS), which is the exclusive OR (XOR) of all the characters in the packet, excluding the checksum itself.

The following additional components are mentioned in the discussion of the SAS functions that support DataMyte data collection:

- start-of-text (STX) character, which is always CTRL-B ('02'x)
- end-of-transmission (EOT) character, which is always CTRL-D ('04'x).

For more information about DataMyte processing, see your DataMyte documentation.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS[®] Companion for the OS/2[®] Environment, Version 8*, Cary, NC: SAS Institute Inc., 1999. 448 pp.

SAS[®] Companion for the OS/2[®] Environment, Version 8

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-521-3

All rights reserved. Produced in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

U.S. Government Restricted Rights Notice. Use, duplication, or disclosure of the software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

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

1st printing, October 1999

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.

IBM[®] and OS/2[®] are registered trademarks or trademarks of International Business Machines Corporation. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The Institute is a private company devoted to the support and further development of its software and related services.