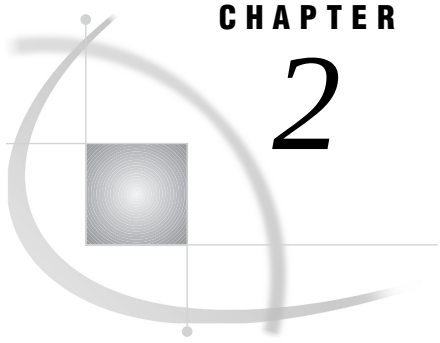


CHAPTER

2

Using the SAS System under OpenVMS

Introduction

Under OpenVMS, you can use any of the following methods to run the SAS System:

- SAS Explorer (see “Using SAS with the SAS Explorer Window” on page 20)
- SAS windowing environment (see “Using SAS with the Windowing Environment” on page 22)
- DMSEXP mode (see “Using SAS in the DMSEXP Mode” on page 24)
- batch mode (see “Using SAS in Batch Mode” on page 25)
- interactive line mode (see “Using SAS in Interactive Line Mode” on page 27)
- noninteractive mode (see “Using SAS in Noninteractive Mode” on page 28).

For additional information about these modes, see *SAS Language Reference: Concepts* and the SAS online Help.

Note: You can also run SAS in a SPAWN/NOWAIT subprocess or in a detached process. SPAWN allows you to use the SAS windowing environment. However, the detached process method is similar to batch mode for queues. For details about these methods, see “Running SAS in a SPAWN/NOWAIT Subprocess” on page 30 and “Running SAS in a Detached Process” on page 30. △

This section also provides information about several other aspects of using the SAS System in the OpenVMS operating environment.

Starting a SAS Session

Regardless of which mode of operation you use for running SAS, you will need to ask your system manager what the *SAS command* (the command that invokes SAS) is at your site. At many sites, the SAS command is simply **SAS**, but a different command may have been defined during the SAS installation process at your site.

Note: The examples in this section use **SAS8** as the SAS command. △

Also ask your system manager which interface or mode of operation is the default when you enter the SAS command.

When you invoke SAS, you can specify system options either when you issue the SAS command or in a configuration file:

```
$ SAS8/FULLSTIMER/PRINT=SYS$LOGIN:TEST.OUT
```

For details, see "Specifying System Options in the SAS Command" on page 45 and "Configuration Files" on page 46.

For more information about SAS system options, see Chapter 18, "System Options," on page 387.

If no system options are specified in the SAS command, a configuration file, an autoexec file, or the VMS_SAS_OPTIONS DCL symbol, then the default system options that are shipped with the SAS System are in effect. However, your system manager may have overridden the default options; ask your system manager for details about the default options at your site.

X Command Line Options

When you invoke some X clients, such as the SAS System, you can use command line options that are passed to the X Window System. In general, you should specify X Window System options after SAS options on the command line with the /XRES option. For example,

```
SAS8/xres="-display wizard:0.0"
```

The X command line options and their values must be enclosed in either single or double quotation marks. If the first blank-separated element of a /XRES argument list does not begin with a hyphen (-), it is assumed to be the name to use for the application instance.

The following list describes the X command line options that are available when you invoke a SAS session from the command prompt:

-display *host:server.screen*
: specifies the name or IP address of the terminal on which you want to display the SAS session. For example, if your display node is wizard, you might enter

```
-display wizard:0.0
```

or

```
-display 10.22.1.1:0
```

-name *instance-name*
: reads the resources in your SAS resource file that begin with the value specified for *instance-name*. For example, **-name MYSAS** reads the resources that begin with **MYSAS** such as

```
MYSAS.dmsfont: Cour14
MYSAS.defaultToolbox: True
```

You can also specify the value for *instance-name* without putting **-name** in the /XRES option if

```
-name
```

is the first item in the quotation marks. For example,

```
SAS8/xres="SAS"
```

is the same as

```
SAS8/xres="-name SAS"
```

-title *string*
specifies the title up to six characters long for your SAS session window. To use multiple words in the title, enclose the words in single or double quotes. For example, **-title MYSAS** produces **MYSAS:Explorer** in the title bar of the Explorer window.

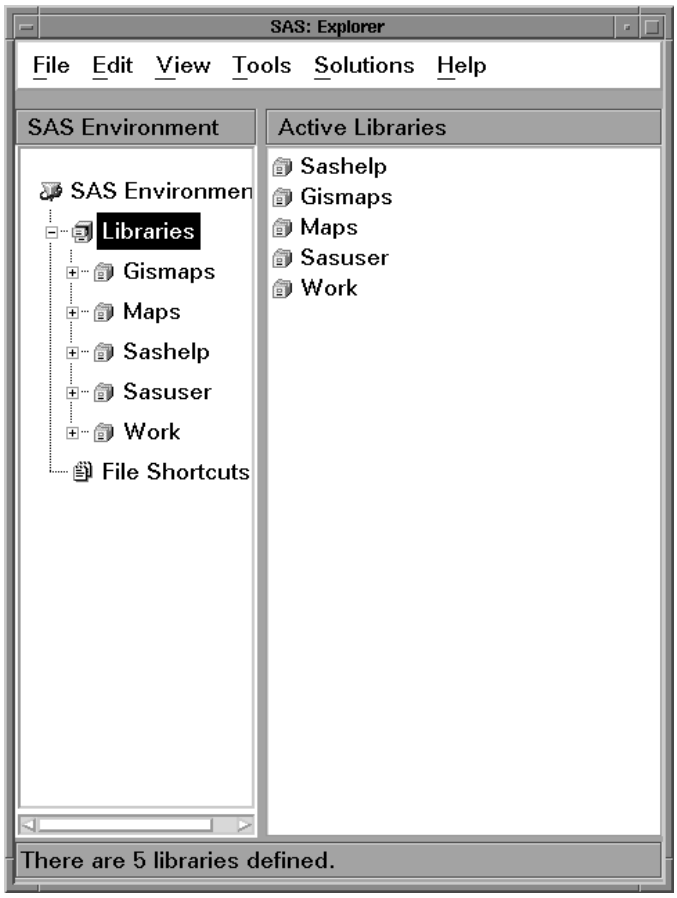
-xrm *string*
specifies a resource to override any defaults. For example, the following resource turns off the Confirm dialog box when you exit SAS:

```
-xrm 'SAS.confirmSASExit: False'
```

Using SAS with the SAS Explorer Window

In Version 8 of the SAS System under OpenVMS, the Explorer window is available on any display device that uses the Motif interface.

Display 2.1 SAS Explorer Window



For information about Motif, see your Motif documentation. The SAS Explorer window is not the default for the SAS System in the OpenVMS operating environment.

Invoking SAS with the Explorer Window

If you have the hardware and software to run the Motif interface, you can use any of the following methods to invoke a SAS process with the Explorer window:

- the SAS command plus any appropriate system options
- the Applications menu of the Motif Session Manager
- a command procedure (.COM) file

Invoking SAS with the SAS Command

If the SAS Explorer window is the default for running SAS at your site, then all you need to do is enter the SAS command (or whatever command your site has chosen) at the DCL prompt:

```
$ SAS8
```

If the SAS Explorer window is not the default, then use the EXPLORER system option to specify the Explorer window:

```
$ SAS8/EXPLORER
```

As explained in "Starting a SAS Session" on page 19, you can also specify other system options when you invoke SAS in this manner.

To specify a display node, use the SET DISPLAY DCL command. For example, if you want to invoke SAS with the Explorer window and to display the windows on node MYNODE running TCP/IP transport protocol, type the following:

```
$ SET DISPLAY/CREATE/NODE=MYNODE/TRANS=TCPIP
$ SAS8/EXPLORER
```

For more information about the SET DISPLAY DCL command, see the OpenVMS online help. For more information about logical names, refer to *OpenVMS User's Manual*.

Invoking SAS from Motif Session Manager

You can also invoke SAS with the SAS Explorer window from the **Applications** menu of the Motif Session Manager. To add SAS to the **Applications** menu (if it is not already there), follow these steps:

1. From the **Options** menu, select **Menus**.
2. Select **Applications** from the **Menu Names** list.
3. Enter a menu item name, such as **MySAS**, and define SAS8/EXPLORER (along with the appropriate command-line qualifiers) as the DCL command.
4. Add the menu definition to the **Item Names** list by clicking on the up arrow.
5. Add the new item name to the existing list of applications by clicking on the left arrow.
6. Click on Apply and then click on Cancel in the Menus dialog window.
7. Select your menu command from the **Applications** menu to invoke the SAS System.

For more information about Session Manager, see the documentation for your Motif interface.

Invoking SAS from a Command Procedure File

To invoke SAS with the Explorer window from a command procedure (.COM) file, place the following commands in the .COM file:

```
$ DEASSIGN SYS$INPUT
$ SAS8/EXPLORER
```

The DEASSIGN command prevents OpenVMS from looking for program input in the location defined by the SYS$INPUT logical name, and it enables SAS to be initialized with the Explorer window.

If the .COM file is named MYSAS.COM, then you would execute the file as follows:

```
$ @MYSAS
```

For more information about command procedure files, see "Command Procedures" on page 13 and *OpenVMS User's Manual*.

Using SAS with the Windowing Environment

The *SAS windowing environment* is an interactive windowing environment for editing, running, and viewing SAS System source code and output.

In Version 8 of the SAS System under OpenVMS, the SAS windowing environment interface is available on any display device that uses the Motif interface.

Invoking SAS with the Windowing Environment Interface

If you have the hardware and software to run the Motif interface, you can use any of the following methods to invoke a SAS process with the SAS windowing environment interface:

- SAS command plus any appropriate system options
- Applications menu of the Motif Session Manager
- command procedure (.COM) file.

Invoking SAS in the windowing environment mode opens the three programming windows: Program Editor, Log, and List.

Invoking SAS with the SAS Command

The SAS windowing environment is the default for running SAS under OpenVMS. Therefore, to invoke SAS all you need to do is enter the SAS command (or whatever command your site has chosen) at the DCL prompt:

```
$ SAS8
```

If the SAS windowing environment is *not* the default, then use the SAS windowing environment system option to specify the SAS windowing environment interface:

```
$ SAS8/DMS
```

As explained in "Starting a SAS Session" on page 19, you can also specify other system options when you invoke SAS in this manner.

To specify a display node, use the SET DISPLAY DCL command. For example, if you want to invoke SAS with the SAS windowing environment interface and to display the windows on node MYNODE running TCP/IP transport protocol, type the following:

```
$ SET DISPLAY/CREATE/NODE=MYNODE/TRANS=TCPIP
$ SAS8/DMS
```

For more information about the SET DISPLAY DCL command, see the OpenVMS online help and *OpenVMS User's Manual*.

Invoking SAS from Motif Session Manager

You can also invoke SAS with the SAS windowing environment interface from the **Applications** menu of the Motif Session Manager. To add SAS to the **Applications** menu (if it is not already there), perform the following steps:

1. From the **Options** menu, select **Menus**.
2. Select **Applications** from the **Menu Names** list.
3. Enter a menu item name, such as **MySAS**, and define SAS8/DMS (along with the appropriate command-line qualifiers) as the DCL command.
4. Add the menu definition to the **Item Names** list by clicking on the up arrow.
5. Add the new item name to the existing list of applications by clicking on the left arrow.
6. Click on Apply and then click on Cancel in the Menus dialog window.
7. Select your menu command from the **Applications** menu to invoke the SAS System.

For more information about Motif Session Manager, see the documentation for your Motif interface.

Invoking SAS from a Command Procedure File

To invoke SAS with the SAS windowing environment interface from a command procedure (.COM) file, place the following commands in the .COM file:

```
$ DEASSIGN SYS$INPUT
$ SAS8/DMS
```

The DEASSIGN command prevents OpenVMS from looking for program input in the location defined by the SYS$INPUT logical name, and it enables SAS to be initialized with the SAS windowing environment interface.

If the .COM file is named MYSAS.COM, then you would execute the file as follows:

```
$ @MYSAS
```

For more information about command procedure files, see “Command Procedures” on page 13 and *OpenVMS User's Manual*.

Using SAS in the DMSEXP Mode

In Version 8 of the SAS System under OpenVMS, the DMSEXP mode is available on any workstation that uses the Motif interface.

Invoking SAS in the DMSEXP Mode

If you have the hardware and software to run the Motif interface, you can use any of the following methods to invoke a SAS process in the DMSEXP mode:

- SAS command plus any appropriate system options
- Applications menu of the Motif Session Manager
- command procedure (.COM) file.

Invoking SAS with the SAS Command

If the DMSEXP mode is the default for running SAS at your site, then all you need to do is enter the SAS command (or whatever command your site has chosen) at the DCL prompt:

```
$ SAS8
```

If the DMSEXP mode is *not* the default, then use the DMSEXP system option to specify the DMSEXP mode:

```
$ SAS8/DMSEXP
```

As explained in “Starting a SAS Session” on page 19, you can also specify other system options when you invoke SAS in this manner.

To specify a display node, use the SET DISPLAY DCL command. For example, if you want to invoke SAS in the DMSEXP mode and to display the windows on node MYNODE running TCP/IP transport protocol, type the following:

```
$ SET DISPLAY/CREATE/NODE=MYNODE/TRANS=TCPIP
$ SAS8/DMSEXP
```

For more information about the SET DISPLAY DCL command, see the OpenVMS online help and *OpenVMS User's Manual*.

Invoking SAS from Motif Session Manager

You can also invoke SAS in the DMSEXP mode from the **Applications** menu of the Motif Session Manager. To add SAS to the **Applications** menu (if it is not already there), perform the following steps:

1. From the **Options** menu, select **Menus**.
2. Select **Applications** from the **Menu Names** list.
3. Enter a menu item name, such as **MySAS**, and define SAS8/DMSEXP (along with the appropriate command-line qualifiers) as the DCL command.
4. Add the menu definition to the **Item Names** list by clicking on the up arrow.
5. Add the new item name to the existing list of applications by clicking on the left arrow.
6. Click on Apply and then click on Cancel in the Menus dialog window.
7. Select your menu command from the **Applications** menu to invoke the SAS System.

Invoking SAS from a Command Procedure File

To invoke SAS in the DMSEXP mode from a command procedure (.COM) file, place the following commands in the .COM file:

```
$ DEASSIGN SYS$INPUT
$ SAS8/DMSEXP
```

The DEASSIGN command prevents OpenVMS from looking for program input in the location defined by the SYS$INPUT logical name, and it enables SAS to be initialized in the DMSEXP mode.

If the .COM file is name MYSAS.COM, then you would execute the file as follows:

```
$ @MYSAS
```

For more information about command procedure files, see "Command Procedures" on page 13 and *OpenVMS User's Manual*.

Using SAS in Batch Mode

SAS *batch mode* is equivalent to OpenVMS batch mode. It is useful for SAS programs that require large amounts of time and memory. In batch mode, you submit a job to an OpenVMS batch queue, and the job awaits execution along with batch jobs that have been submitted by other OpenVMS users. The amount of time before your job executes depends on how many other jobs are in the input queue, on what priority the operating environment has assigned to your job, and how your batch queue(s) is configured.

You can use your terminal for other tasks while the job awaits execution, but you cannot change the submitted job in any way until after it executes.

Usually, the first step in executing a program in batch mode is to prepare two types of files:

command procedure file
contains the DCL commands that are used to set up the SAS environment. For example, it may include commands that do the following:

- define OpenVMS logical names
- set your default directory to access your data
- invoke the SAS System with the appropriate SAS system options.

program file
contains the SAS program that you want to execute. The name of this file can be included in the text of the command procedure file, or it can be passed as a parameter to the command procedure file. See the examples in "Examples of Batch Job Files" on page 26.

Examples of Batch Job Files

Following are three examples of using command procedure files and program files in batch mode.

Example 1: Separate Command Procedure and Program Files

In this example, we first create a file called MYPROG.SAS that contains the following simple SAS program:

```
libname in 'disk:[directory]';
proc print data=in.mydata;
   title 'A Simple SAS Program';
run;
```

Next we create a command procedure file called CONTROL.COM that contains the following DCL command:

```
$ SAS8/LINESIZE=76/NODATE [HOME.SUBDIR]MYPROG.SAS
```

CAUTION:
Do not give your SAS program and the command procedure the same name. Giving them the same name causes confusion when both the OpenVMS log and the SAS log are created. The OpenVMS log is created first (for example, MYPROG.LOG;1) and the SAS log is created second (MYPROG.LOG;2). If your OpenVMS system has been set up to keep only one version of a file, then the OpenVMS batch log will be overwritten by the SAS log. △

To submit the SAS job, we enter the following command at the DCL prompt:

```
$ SUBMIT/NOTIFY CONTROL.COM
```

The job is placed in the default batch queue, and the terminal session is available for other work. You will be notified by the operating environment when your batch job has completed.

Example 2: Passing the Name of the Program File as a Parameter

You can make your command procedure file more generic and pass the name of the SAS program as a parameter. This is helpful if you want to execute several programs in the same environment. To do this, you would modify the command procedure file from Example 1 as follows:

```
$ SAS8/LINESIZE=76/NODATE  'P1'
```

The 'P1' at the end of the SAS command line is the placeholder for the parameter that you are going to pass to the command.

You could then submit a program called MYPROG.SAS by executing the following command:

```
$ SUBMIT/NOTIFY/PARAMETER=("MYPROG.SAS")-
_$ CONTROL.COM
```

Example 3: Including the Program File in the Command Procedure File

A third alternative is to include the SAS program in the same file as the control commands. In a batch environment, the OpenVMS system assumes that the input source is the command procedure file that is executing. This input source is named by the OpenVMS logical name SYS$INPUT. To combine the control commands and the SAS program in the same file, create a command file that contains the following lines:

```
$ SAS8/LINESIZE=76/NODATE-
   /PRINT=DISK:[DIRECTORY]MYPROG.LIS-
   /LOG=DISK:[DIRECTORY]MYPROG.LOG SYS$INPUT
libname in 'disk:[directory]';
proc print data=in.mydata;
   title 'A Simple SAS Program';
run;

endsas;
```

The hyphen at the end of the first line of the SAS command indicates that the command continues on the next line. Designating SYS$INPUT as your input file tells SAS that your input will be included in the text of the command procedure file. Submit this job as you would any other batch job at your site.

Using SAS in Interactive Line Mode

In SAS *interactive line mode*, you enter SAS statements line by line in response to line prompts that are issued by the SAS System. The SAS session retains exclusive use of the terminal or terminal window.

Invoking SAS in Interactive Line Mode

To invoke SAS in interactive line mode, enter the following command at the DCL prompt:

```
$ SAS8/NODMS
```

(If your system manager has set up the SAS environment so that the NODMS system option is the default, you can omit the NODMS option.)

As explained in "Starting a SAS Session" on page 19, you can also specify other system options when you invoke SAS in this manner.

When SAS prompts you with the 1? prompt, enter your SAS statements. By default, both the SAS log and the procedure output files (if any) appear on your display as each step executes.

Recalling SAS Statements with CTRL-B and the Arrow Keys

Under OpenVMS you can recall SAS statements by using the CTRL-B key combination and your up arrow and down arrow keys.

Both CTRL-B and the up arrow key move you backward through SAS statements that you entered previously. For example, if you have entered four lines of code and want to recall the first line, press CTRL-B or the up arrow key four times. Each time you press CTRL-B or the up arrow key, the previous line appears after the SAS prompt. When the correct line appears, press RETURN to submit the code.

If you accidentally miss the line that you wanted, then use the down arrow key to move forward through the list of previously entered commands. Each time you press the down arrow key, the next statement appears after the SAS prompt.

Saving SAS Statements

Interactive line mode is most effective when you use the %INCLUDE statement to bring in most of your SAS statements. Using this method, the programs that you enter are not long, and you do not need to save your SAS statements. However, if you want to save your program, the best method is to write your SAS log to an OpenVMS file.

One easy way to save your SAS statements is to use the PRINTTO procedure followed by the %LIST statement after you have entered your program statements. For example:

```
. . . program statements . . .
14? filename mylog '[sasdemo]program.log';
15? proc printto log=mylog;
16? run;
17? %list;
```

This program gives you a listing of every line you have entered during your current SAS session. The lines are saved in an OpenVMS file that is referred to by the fileref MYLOG, which was assigned in the FILENAME statement in line 14.

Note: To redirect the log to your display, enter the following PROC PRINTTO statement with no options:

```
proc printto;
run;
```

△

Ending Your SAS Session

The ENDSAS statement terminates SAS interactive line mode:

```
endsas;
```

Then OpenVMS prompts you for a DCL command.

Note: Like all SAS statements, the ENDSAS statement must be followed by a semicolon (;). △

Using SAS in Noninteractive Mode

In SAS *noninteractive mode*, OpenVMS retains exclusive use of the terminal as the noninteractive SAS job executes. However, if your SAS program invokes windowing

procedures, you can interact with the program during program execution. Your SAS program is read from a file that you specify by including the filename in the SAS command.

Note: Noninteractive mode is similar to batch mode: statements are usually not expected to come from the terminal, and the SAS log and procedure output are routed to files with .LOG and .LIS file types by default. △

Invoking SAS in Noninteractive Mode

To invoke SAS in noninteractive mode, enter the SAS command followed by the name of the SAS program file that you want to execute. For example, suppose you have stored SAS statements and data lines in a program file named [HOME.SUBDIR]MYPROG.SAS. At the DCL prompt, enter the SAS command and the name of the file as follows:

```
$ SAS8 [HOME.SUBDIR]MYPROG
```

You do not need to include the file type in the filename because the SAS command uses the .SAS file type by default. If [HOME.SUBDIR] is your current default directory, then you can omit the directory name from the file specification as follows:

```
$ SAS8 MYPROG
```

In either case, SAS executes the statements in MYPROG.SAS and creates two files in the default directory: MYPROG.LOG contains the SAS log output and MYPROG.LIS contains the output from any SAS procedure that produces output.

To print one or both of these files, use the PRINT DCL command. To view these files at your terminal, use the EDIT or TYPE command. Note that if you use the TYPE command to list a SAS log that contains errors, overprinting obscures the line containing the error unless the OVP system option was set to NOOVP during your SAS session. (For a description of the OVP system option, see *SAS Language Reference: Dictionary*.)

Recalling SAS Statements

In noninteractive mode, the %INCLUDE statement serves two purposes. You can use it to include SAS statements that are stored in an external file, or you can issue the following form of the statement to allow input from your current terminal:

```
%include *;
```

Program execution pauses as you are prompted for input from the terminal. The prompt is a line number and an asterisk (*). Although this input method simulates SAS interactive line mode (because you are prompted for statements line by line), the statements are not interpreted line by line. This means that syntax errors are not detected until you terminate input and return to noninteractive processing.

To terminate input and resume noninteractive processing, press CTRL-Z or enter a %RUN statement:

```
%run;
```

Note: Like all SAS statements, the %RUN statement must be followed by a semicolon (;). △

Running SAS in a SPAWN/NOWAIT Subprocess

The SPAWN= system option enables you to run the SAS System in a SPAWN/NOWAIT subprocess. This method of running SAS is similar to batch mode if you do not specify /DMS or /EXPLORER. The only difference is that in batch mode, SAS runs in its own process rather than in a spawned subprocess.

When you specify SPAWN=NOWAIT in the SAS command and do not specify /DMS or /EXPLORER, the SAS System assumes the terminal cannot be used to take input from the user. Therefore, the SAS System will not run in interactive line mode or in the windowing environment using the terminal-based windowing environment if you do not specify /DMS or /EXPLORER. You can, however, run the SAS System in batch mode, or invoke SAS under the Motif interface, as shown in the following SAS commands:

```
$ SPAWN/NOWAIT SAS8/SPAWN=NOWAIT MYFILE.SAS

$ SPAWN/NOWAIT SAS8/SPAWN=NOWAIT/EXPLORER

$ SPAWN/NOWAIT SAS8/SPAWN=NOWAIT/DMS
```

Attention-handling is disabled if you are running the SAS System with SPAWN=NOWAIT in effect. Thus, to terminate the subprocess running the SAS System, use the STOP DCL command.

For details about invoking the SPAWN= system option, see “SPAWN=” on page 440.

Running SAS in a Detached Process

Unlike a spawned process, a detached process is independent of other processes; it is not linked to an OpenVMS parent process. A batch job is one type of detached process. To avoid the restrictions of batch processes, or to free up your DECterm window and still invoke SAS interactively or noninteractively, you can submit a SAS job to run in a detached process.

The following DCL command procedure file, DETACH.COM, invokes SAS with the SAS windowing environment interface in a detached process. (The numbered lines are explained following the code.)

```
   $! DETACH.COM
   $ SET NOON
❶$ DEFINE SYS$LOGIN DISK:[HOMEDIR]
   $ SET DEFAULT SYS$LOGIN
   $ @SYS$LOGIN:LOGIN.COM
   $ DEFINE SAS$NET_TYPE XCLIENT
❷$ @DUA1:[SAS8.TOOLS]SAS8.COM
   $ SAS8:==$SAS$ROOT:[IMAGE]SAS8.EXE
   $ SAS8/DMS
   $ EXIT
```

1 Replace **DISK:[HOMEDIR]** with your own disk.

2 Replace **DUA1:[SAS8.TOOLS]** with your own location for the SAS System. Leave **SAS8.COM** as it is. (The SAS8.COM file defines OpenVMS logical names that the SAS System uses. For more information about the SAS8.COM file, see “OpenVMS Logical Names That SAS Uses” on page 52.)

To execute the DETACH.COM file, you would use a command like the following:

```
$ RUN/DETACHED/INPUT=DETACH.COM-
_$ /OUTPUT=DETACH.LOG-
_$ SYS$SYSTEM:LOGINOUT.EXE
```

Note: If your LOGIN.COM does not include a SET DISPLAY command, you must include one in your DCL command procedure. △

Interrupting a SAS Session

If you are running SAS with the SAS Explorer window in the SAS windowing environment, in interactive line mode, or in noninteractive mode, then you can interrupt your SAS session by pressing CTRL-Y. This is called an attention sequence. After you press CTRL-Y, the SAS System will issue you a message. The content of the message depends on the interface you are using and on what task SAS is performing at the time of the interrupt.

In line mode, you will receive the following message:

```
Select:
  1. Line Mode Process
  2. DATASTEP
  C. Cancel
  T. Terminate System
```

In the SAS windowing environment, you will receive the following message:

```
Select:
  1. DMS Process
  2. Language Processor
  3. DATASTEP
  C. Cancel
  T. Terminate System
```

In the windowing environment if you select **1**, you will see the following instructions:

```
Press Y to terminate this SAS process,
      N to continue.
```

If you type **Y**, you will terminate the SAS process. If you type **N**, the process will continue executing.

In line mode if you select **1** or if you select **2** in the windowing environment, you will see the following instructions:

```
Press Y to cancel submitted statements,
      N to continue.
```

If you type **Y**, the SAS System cancels the statements that were submitted. If you type **N**, the statements will continue executing.

If you select **DATASTEP**, you will see the following instructions:

```
Press Y to halt data step/proc,
      N to continue.
```

If you type **Y**, the SAS System stops processing the DATA step or procedure. If you type **N**, the DATA step or procedure will continue executing.

If you select **C**, the interruption (begun by pressing CTRL-Y) is cancelled. If you select **T**, your SAS session is terminated.

Usually these messages appear immediately after you press CTRL-Y.

Note: If you have to wait a few seconds for these messages, do not repeatedly press CTRL-Y. If you press CTRL-Y more than once, or if you press CTRL-Y while SAS is in the process of initializing or shutting down, the WORK files may not be deleted. In this case, you must use the CLEANUP tool to delete them. For more information, see "The CLEANUP Tool" on page 35. △

When you end your SAS session with a CTRL-Y, usually all temporary WORK files are properly deleted. In noninteractive mode, if you interrupt a SAS session, the .LOG and .LIS files are retained, up to the point where you pressed CTRL-Y.

Note: In Version 8 of the SAS System, CTRL-Y and CTRL-C function identically. You can follow this same procedure to interrupt a SAS task. △

The WORK Data Library under OpenVMS

Disk space is the aspect of the WORK library that is most likely to require your consideration. If you have many large temporary SAS data sets, or if you use a procedure that has many large utility files (for example, a PROC FREQ step with a complex TABLES statement that you run against a large SAS data set), you may run out of disk space in the WORK library. If you run out of disk space in batch mode, your PROC or DATA step terminates prematurely and issues a message similar to the one shown in Output 2.1 on page 32. In an interactive session, a dialog window asks you to specify what action to take.

Output 2.1 Insufficient WORK Space Message

```
ERROR: Insufficient space in file WORK.DATASET.DATA.
   NOTE: The SAS System stopped processing this step because of errors.
   NOTE: SAS set option OBS=0 and will continue to check statements.
         This may cause NOTE: No observations in data set.
   WARNING: The data set WORK.DATASET may be incomplete.  When this step
            was stopped there were 22360 observations and 4 variables.
   ERROR: Errors printed on page 1.
   NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC 27513
```

Ask your system administrator to increase the disk quota that has been assigned to you, or ask for the disk to be cleaned up. Following are several methods of increasing your WORK space:

- Designate a disk with more space as your WORK library. (See "Changing the Location of the WORK Library" on page 33.)
- Delete each temporary SAS data set as soon as you no longer need it. (See "Deleting Temporary SAS Data Sets" on page 33.)

- Direct the temporary SAS data sets to a different SAS data library so that disk space in the WORK library is conserved for items that must be stored there. (See "Directing Temporary SAS Data Sets to the USER Library" on page 33.)
- Reduce the version limit on the directory in which the WORK library is created.

You can also combine these methods.

Changing the Location of the WORK Library

By default, a subdirectory of the current directory contains the WORK data library. The name of the subdirectory that SAS creates for your WORK data library is SAS$WORK*current-pid*, where *current-pid* is the unique 8-byte process-identification value that is assigned by OpenVMS. By default, this directory is created as a subdirectory of the directory that is referenced by the OpenVMS logical name SAS$WORKROOT. SAS$WORKROOT refers to SYS$DISK:[ ] by default, but your system manager may have redefined it. The default may be changed when the SAS System is installed by your system manager.

You can change the location of the WORK data library either by redefining the OpenVMS logical name SAS$WORKROOT or by specifying the WORK= system option. For example, the following SAS command tells SAS to create the WORK subdirectory in the directory DISK:[XDIR]:

```
$ SAS8/WORK=DISK:[XDIR]
```

When SAS creates the WORK subdirectory, it also creates an OpenVMS process-level logical name, SAS$WORKLIB, that references the WORK data library. You can use this logical name within the SAS session to reference files in the WORK subdirectory. It remains defined after the SAS session terminates.

Deleting Temporary SAS Data Sets

Under OpenVMS, *temporary SAS data set* means a data set that is stored in a temporary SAS work library. That is, you cannot designate the data set itself as temporary, but the data set takes on the attribute of the library in which it is stored.

One simple way to conserve space in the WORK library is to delete each temporary SAS data set with a PROC DATASETS step after you no longer need it. However, there are two problems with this method.

- You can cause errors in a job by deleting a SAS data set before the job is finished with it.
- If you need several very large temporary SAS data sets in your job at the same time, you may run out of space before you reach a point at which you can delete any SAS data sets.

Directing Temporary SAS Data Sets to the USER Library

An alternative to deleting the temporary SAS data sets is to direct them to a different SAS data library. You can use the USER= system option to store temporary data sets in the USER library rather than in the WORK library.

Note: Utility data sets that are created by SAS procedures continue to be stored in the WORK library. However, any data sets that have one-level names and that are created by your SAS programs will be stored in the USER library. △

The following example illustrates the use of the USER= system option. The numbered lines are explained following the code.

```
filename giant 'mydisk:[survey]tvdata.dat';
libname result 'mydisk:[sasdata]';
❶  libname temp 'disk2:[temp]';
❷  options user=temp;
❸  data totalusa;
     infile giant;
     input home_id region income viewers cable;
     if home_id=. then delete;
   run;

❹  proc freq;
     tables region*income*viewers*cable
❺  / noprint out=result.freqdata;
   run;
```

1 The LIBNAME statement associates the libref TEMP with the directory DISK2:[TEMP].

2 In the OPTIONS statement, the USER= system option designates the TEMP libref as the temporary SAS data library. Any data sets that have one-level names and that are created by your SAS program will be stored in this library.

3 A one-level name is used in the DATA statement. When the DATA step is processed, the SAS data set TEMP.TOTALUSA is created.

4 Because the large TOTALUSA data set was directed to the TEMP library, there is more space available in the WORK library for the utility files that the FREQ procedure requires.

5 The SAS data set FREQDATA contains the results of the FREQ procedure. A two-level name is used to store FREQDATA in the permanent SAS data library MYDISK:[SASDATA].

Note: You can also assign the USER libref directly by using the LIBNAME statement as follows:

```
libname user '[mydir]';
```

△

You can specify the USER= system option in the SAS command, as in the following example:

```
$ SAS8/USER=[MYDIR]
```

Note: Unlike the WORK library, when you use the user library to store temporary files, these files are not automatically deleted when the SAS System terminates. △

System Options That Control the WORK Data Library

Two system options control the creation and deletion of the WORK subdirectory. By default, the WORK subdirectory is deleted at the end of each SAS session, and a new one is created at the beginning of the next session. The WORKTERM system option controls the deletion of the WORK subdirectory, and the WORKINIT system option controls the creation of the new subdirectory.

The default value of the WORKINIT system option is WORKINIT, and the SAS System automatically creates a WORK subdirectory as it initializes. If you specify NOWORKINIT, the SAS System looks for an existing WORK subdirectory in the current

directory and uses it, as is, if it exists. If it does not exist, one is created. You can specify the WORKINIT system option in the SAS command or in a configuration file.

If you have logged out of your OpenVMS process since the previous WORK subdirectory was saved, the SAS System cannot find the WORK data library even if you specify NOWORKINIT, so it creates a new WORK data library. This is because your OpenVMS process ID has changed and the subdirectory name includes the OpenVMS process ID. In this case, the old WORK data library still exists (it was not written over), and you can assign a new libref to the old WORK subdirectory (SAS$WORK*old-pid*) after you get into your SAS session and use the files through the new libref. Search the default directory for the old SAS session to find the old WORK subdirectory name. You can use the X statement within your current SAS session to perform this directory search.

The WORKTERM option controls whether the WORK subdirectory is deleted at the end of your SAS session. The default value is WORKTERM. If you specify NOWORKTERM, the WORK subdirectory is not deleted at the end of the session and remains available for use with the next SAS session. Remember, however, that you must specify NOWORKINIT in the next invocation of the SAS System in order to reuse the WORK subdirectory that you saved.

The WORKTERM and WORKINIT system options are portable and are documented in *SAS Language Reference: Dictionary*.

The CLEANUP Tool

Under OpenVMS, the CLEANUP tool is available to conveniently delete the WORK data library. When a SAS session terminates normally, the WORK library is deleted automatically (this is the default action). However, if your SAS session terminates abnormally, the WORK library and the files in it may not be deleted properly. To delete this WORK data library, use the CLEANUP tool.

CAUTION:
Do not use the CLEANUP tool if your default directory is also the default directory for a SAS batch job, and a SAS batch job is currently running. If you inadvertently delete the WORK subdirectory that was created for a batch job, the job abends. Also, never issue the CLEANUP command from an OpenVMS subprocess. △

To access the CLEANUP tool, first create a DCL symbol that points to the executable image. This symbol or foreign command can be added to the SAS8.COM file. The symbol CLEANUP is used in the following example:

```
$ CLEANUP == "$SAS$ROOT:[PROCS]CLEANUP.EXE"
```

Then, issue the CLEANUP command from your DCL prompt to delete WORK data libraries from one or more directories.

The following are some syntax variations to keep in mind:

To clean the current directory of work files, type

```
$ CLEANUP
```

To clean work files from [.TEMP], type

```
$ CLEANUP [.TEMP]
```

To clean work files from an entire directory tree, type

```
$ CLEANUP [...]
```

The CLEANUP command accepts the following qualifiers:

/LOG | /NOLOG
/LOG is the default. This causes the CLEANUP tool to issue a message showing each file as it is deleted, and it shows how many directories were deleted. If you specify /NOLOG, no messages are issued unless the CLEANUP tool encounters an error while trying to delete one or more files.

/CONFIRM | /NOCONFIRM
/CONFIRM is the default. This causes the CLEANUP tool to prompt you for each directory to be deleted. The default is Y for yes if you press RETURN. If you do not want to be prompted, or if you are submitting the command from a batch job, then use /NOCONFIRM.

Note: The /V5 | /NOV5 and /V6 | /NOV6 options for the CLEANUP command are not supported in Version 8 of the SAS System. △

For each WORK data library that exists in the specified directory, the CLEANUP tool issues the question

```
OK to delete
    device:[dir]SAS$WORKnnnnnnnn.DIR;1  [YES]?
```

where ***device*:[*dir*]** is the name of the disk and directory to which you were attached when you invoked the SAS System, and **SAS$WORK*nnnnnnnn*.DIR;1** is the subdirectory that contains the WORK library. YES is the default response, so to execute the CLEANUP command, press the RETURN key. When the CLEANUP tool executes, it lists the names of the files being deleted. If the specified directory contains more than one WORK subdirectory, the CLEANUP tool then issues the previous question for the next WORK data library.

If the CLEANUP tool is not available at your site, contact your SAS Support Consultant.

Issuing DCL Commands during a SAS Session

You can issue DCL commands from within a SAS program by using either the SAS X statement (in any mode of SAS operation) or the SAS X command in interactive and non-interactive mode. Depending on which form of the X statement (or X command) that you use, you can either issue a single DCL command or you can spawn an OpenVMS subprocess from which you can issue multiple DCL commands.

System administrators can specify the NOXCMD option to prevent users from issuing any DCL commands.

You can use the SYSTASK statement to execute DCL commands asynchronously. For more information, see “SYSTASK” on page 380 and “WAITFOR” on page 384.

Note: In general, the X statement and the X command are equivalent. The following discussion illustrates the X statement but points out any differences that would apply to the X command. △

Using the X Statement to Issue a Single DCL Command

Use the following form of the X statement to issue one DCL command:

X <'>*DCL-command* <'>;

The *DCL-command* is passed to the operating environment and executed. If errors occur, the appropriate error messages are displayed.

In interactive line mode, after the command executes you are prompted for another SAS statement. For example, in Display 2.2 on page 37 the X statement is submitted from an interactive line mode session and executes the DIRECTORY DCL command. The contents of the current directory appear on the display, and then the SAS System prompts the user for another SAS statement.

Display 2.2 Issuing DCL Command with the X Statement

In Version 8 of the SAS System, the XLOG option specifies to write the output from the X command to the SAS log. However, XLOG is not the default. When XLOG is not turned on, the output from the X command is displayed in the invocation window. The XCMDWIN option determines whether X command output is displayed in a DECTERM window or in the invocation window. XCMDWIN is on by default. For more details on the XCMDWIN option, see "XCMDWIN" on page 452.

Note: You can also use the VMS function, the CALL SYSTEM routine, or the %SYSEXEC macro statement to issue single DCL commands. For details, see the function "VMS" on page 318, the routine "CALL SYSTEM" on page 278, and "Macro Statements" on page 463. △

How OpenVMS Processes the DCL Command

It is important to understand how OpenVMS processes the DCL command that you issue with the X statement. A DCL command can be executed in either the OpenVMS parent process (the OpenVMS process in which your SAS session is running) or in an OpenVMS subprocess. For some DCL commands, such as DIR or COPY, it makes little difference whether it executes in the parent process or a subprocess. However, for DCL commands such as DEFINE and MOUNT, the process in which they are executed is significant. The following DCL commands are executed in the parent process:

ALLOCATE

ASSIGN

ATTACH

DEALLOCATE

DEASSIGN

DEFINE

MOUNT

SET DEFAULT

The results of these DCL commands (for example, OpenVMS logical names, tape mounts, and so on) are available to the OpenVMS parent process. All other DCL commands are executed in a subprocess. Therefore, when you use an X statement to submit these commands from SAS, their results are not available to the OpenVMS

parent process in which SAS is running. For example, the following DCL command has no effect in the OpenVMS parent process:

```
x 'set protection=o:d/default';
```

This command is executed in a subprocess, so the default protection for the OpenVMS process in which SAS is running does not change.

The XTIMEOUT= system option determines whether a subprocess remains in existence after a DCL command has been executed and control has been returned to SAS. (See the system option "XTIMEOUT=" on page 458.) By default, the same subprocess is used during the entire SAS session. However, when any of the commands in the previous list are executed, the following actions occur regardless of the value of the XTIMEOUT= system option:

- The subprocess is deleted.
- A new subprocess is started when the next X statement is issued.

These actions ensure that the subprocess always accurately reflects any OpenVMS logical names or defaults that were set in the parent process.

Executing a DCL Command Using Procedure Syntax

You can also use procedure syntax to issue a single DCL command. However, the X command and the X statement are more versatile because they each allow you to specify parameters and qualifiers for a DCL command; with procedure syntax, you cannot specify parameters or qualifiers.

When you ask the SAS System to run a procedure—for example, a procedure named MYPROC—SAS first tries to load the Version 8 procedure named MYPROC. If the procedure does not exist, then SAS searches for either an image named MYPROC.EXE in your current directory or the OpenVMS logical name MYPROC. If the image MYPROC.EXE is found in your current directory, then SAS issues the MYPROC DCL command via a subprocess. If the OpenVMS logical name exists for MYPROC, then SAS translates the logical name and searches for the image that it points to. If it finds the image, it issues the MYPROC DCL command via a subprocess. If it does not find the image in your current directory, and if no OpenVMS logical name is assigned to the image or if the logical name does not point to an image that exists, then you receive the following error message:

```
ERROR: Procedure MYPROC not found.
```

In this example, note that if the SAS System issues the MYPROC command, you must either define a symbol called MYPROC or have a DCL verb defined for MYPROC. Otherwise, you receive the error message

```
%DCL-W-IVVERB, unrecognized command verb -
               check validity and spelling.
```

The most useful application of using procedure syntax to issue a DCL command is executing a stand-alone image. Suppose you have a program called CALC.EXE in your current directory which performs arithmetic calculations, and you want to run it from within your SAS session. First, define a symbol with the same name as the image you want to invoke. In this case, you define CALC as the following:

```
$ CALC := RUN MYDISK$:[MYDIR]CALC.EXE
```

Then, from your SAS session, run your program using a procedure statement. The following is an example:

```
$ SAS8
   . . . Log notes . . .
```

```
1? proc calc;
2? run;
    Input the calculation: 2+2=
        4.000000
3? endsas;
```

Now suppose that the CALC.EXE image is located in a different directory. The symbol for CALC must still be defined, as in the previous example, but you also need an OpenVMS logical name that points to the actual file. The following is an example:

```
$ DEFINE CALC MYDISK$:[MYPLAYPEN.SUBDIR]CALC.EXE
```

Now SAS can find the image that is defined by the OpenVMS logical name CALC and can issue the CALC DCL command.

Note: You can accomplish the same thing by issuing the following X command:

```
$ SAS8
   . . . Log notes . . .
  1? x 'run calc.exe';
       Input the calculation: 2+2=
               4.000000
  2? endsas;
```

△

Using the X Statement to Issue Several DCL Commands

You can use the X statement to spawn an OpenVMS subprocess from which you can enter multiple DCL commands. In interactive line mode or noninteractive mode, use either of the following forms of the X statement:

- **x '';**
- **x;**

These statements make it easy for you to manage files and devices from within your SAS session. When you use these forms of the X statement, you receive the following message:

```
Type LOGOFF to return to SAS.
```

During the subprocess, you enter DCL commands in response to the following prompt:

```
SAS_$
```

In Version 8, the SAS_$ prompt appears in the terminal window from where you invoked SAS.

In Version 8, the XCMDWIN option specifies whether to create a DECTERM window from which the spawned subprocess reads its input and writes it output. This option applies only in windowing mode. For more information about the XCMDWIN option, see "XCMDWIN" on page 452.

Note: If you issue DCL commands that affect or describe process attributes, be aware that these commands pertain to the subprocess, not to the process in which SAS is running. For example, if you define OpenVMS logical names within the subprocess, these logical names are not available to the OpenVMS parent process in which SAS is running. Therefore, it is usually best to define logical names with the form of the X statement described in "Using the X Statement to Issue a Single DCL Command" on page 36. △

To return to the SAS session, enter LOGOFF in response to the SAS_$ prompt.

SAS System Options That Affect Subprocesses

The following SAS system options affect the subprocess environment that is created by an X statement or X command:

XCMDWIN
in windowing mode, determines whether a separate DECTERM window is created. For details about the XCMDWIN option, see "XCMDWIN" on page 452.

XKEYPAD
affects the keypad definitions that are used in the subprocess. For more information, see the system option "XKEYPAD" on page 453.

XLOGICAL
affects the logical names that are used in the subprocess. For more information, see the system option "XLOGICAL" on page 454.

XOUTPUT
affects the display of output from the X command. For more information, see the system option "XOUTPUT" on page 455.

XSYMBOL
affects the Command Line Interpreter (CLI) symbols that are used in the subprocess. For more information, see the system option "XSYMBOL" on page 457.

XTIMEOUT=
affects how long a subprocess exists after control is returned to the SAS System. For more information, see the system option "XTIMEOUT=" on page 458.

Note: These options take effect only after a subprocess is spawned. △

Issuing OpenVMS Functions from Captive Accounts

If you run SAS from a captive OpenVMS account, then you cannot access the DCL command level. Nevertheless, you can use the X statement or the X command to issue a number of general-purpose OpenVMS functions.

A *captive OpenVMS account* is under the control of the login command procedure. It cannot access the DCL command level, nor can it spawn a subprocess. Captive accounts are set up in the AUTHORIZE system utility by specifying the option /FLAGS=CAPTIVE.

As explained in "Issuing DCL Commands during a SAS Session" on page 36, if you run SAS from a *noncaptive* account, you can issue DCL commands by means of the SAS X command, X statement, VMS function, or CALL SYSTEM routine. For most DCL commands, each of these methods spawns an OpenVMS subprocess, and the subprocess executes the DCL command that you specified. (The exceptions are the commands listed in "How OpenVMS Processes the DCL Command" on page 37, which execute in the OpenVMS parent process. Also note that only the X statement and the X command can be used to spawn a subprocess from which you can issue multiple DCL commands. The other methods can be used only to issue single DCL commands.)

In general, if you issue one of these SAS commands, statements, functions, or CALL routines from a *captive* account, a message informs you that a subprocess cannot be spawned from a captive account; the command requested is denied, and control of the process is returned to SAS.

However, several general-purpose OpenVMS functions can be called from within the same process that is running the SAS System. Therefore, even if you are running SAS from a captive account, you can use the FINDFILE function (for example) to see a directory listing of the files in a certain directory. The results are the same as if you had issued an X command from a noncaptive account. Similarly, you can use the DELETE function from a captive account to delete a file from a particular directory structure, and you can use the RENAME function to rename a file.

The following OpenVMS functions are available in the SAS System and can be called from a captive account:

DELETE
: deletes a file. For more information, see the function “DELETE” on page 280.

FINDFILE
: searches a directory for a file. For more information, see the function “FINDFILE” on page 290.

GETDVI
: returns a specified item of information from a device. For more information, see the function “GETDVI” on page 294.

GETJPI
: retrieves job-process information. For more information, see the function “GETJPI” on page 295.

GETLOG
: returns the value of a DCL logical name. For more information, see the function “GETLOG” on page 296.

GETMSG
: translates an OpenVMS error code into text. For more information, see the function “GETMSG” on page 297.

GETQUOTA
: retrieves disk quota information. For more information, see the function “GETQUOTA” on page 298.

GETSYM
: returns the value of a DCL symbol. For more information, see the function “GETSYM” on page 299.

GETTERM
: returns the characteristics of your terminal device. For more information, see the function “GETTERM” on page 300.

PUTSYM
: creates a DCL symbol in your process. For more information, see the function “PUTSYM” on page 305.

RENAME
: renames a file. For more information, see the function “RENAME” on page 307.

SETTERM
: modifies a characteristic of your terminal device. For more information, see the function “SETTERM” on page 307.

TERMIN
: allows simple input from SYS$INPUT. For more information, see the function “TERMIN” on page 311.

TERMOUT
: allows simple output to SYS$OUTPUT. For more information, see the function “TERMOUT” on page 312.

TTCLOSE
closes a channel that was previously assigned by TTOPEN. For more information, see the function “TTCLOSE” on page 313.

TTCONTRL
modifies characteristics of a channel that was previously assigned by TTOPEN. For more information, see the function “TTCONTRL” on page 313.

TTOPEN
assigns an I/O channel to a terminal. For more information, see the function “TTOPEN” on page 314.

TTREAD
reads characters from the channel that was assigned by TTOPEN. For more information, see the function “TTREAD” on page 316

TTWRITE
writes characters to the channel that was assigned by TTOPEN. For more information, see the function “TTWRITE” on page 317.

Misuse of these functions can occur only if your system has insufficient file-protection and directory-protection schemes. The SAS System honors all protection schemes. For example, if you cannot delete a file from a noncaptive account, then you cannot delete that file from a captive account either.

Note: System administrators can restrict users from calling the above functions from captive accounts by either renaming or deleting the appropriate executables or by making the NOXCMD option the default. The executables that are associated with the above functions are stored in the directory SAS$EXTENSION:[LOAD]. △

Identifying and Resolving Problems

The SAS Institute Web pages contain a great deal of information that is useful for problem-solving and for other purposes. For example, in the “Technical Support” section, you will find the following resources:

SAS Notes
information about reported problems, fixes, and undocumented features of SAS Software. (This information is updated regularly on the SAS Web pages. The SAS Notes are also distributed on all product installation CD-ROMs or tapes and are available on a separate CD-ROM or tape by request. SAS Installation Coordinators can contact SAS Institute’s Distribution Center to request a current copy of the notes.)

Frequently Asked Questions (FAQs)
answers to the questions that the SAS Institute Technical Support staff are most frequently asked by SAS software users.

Sample Programs
a collection of thousands of SAS programs that demonstrate different features of SAS Software and that illustrate how to use SAS software to solve application problems.

You will also find information about how to report problems via e-mail.

Determining the Completion Status of a SAS Job

Under OpenVMS, three symbols are set at SAS termination that indicate the success or failure of the SAS job: SAS$STATUS, $SEVERITY, and $STATUS.

- SAS$STATUS indicates the final state of the SAS session. A value of 0 indicates normal termination. If any of the following versions of the SAS ABORT statement are used, then SAS$STATUS is set to the value *n*:

  ```
  abort n;
  abort return n;
  abort abend n;
  ```

  where *n* can range from –2,147,483,648 to 2,147,483,647.

  If you issue these statements without specifying *n*, then SAS$STATUS is set to the following values:

  abort;
  sets SAS$STATUS to 12.

  abort return;
  sets SAS$STATUS to 12.

  abort abend;
  sets SAS$STATUS to 999.

  If a fatal error occurs and the SAS session does not terminate normally, then SAS$STATUS is set to either 999 or 998, depending on the type of internal error.

  For more information about the ABORT statement, see “ABORT” on page 353 and *SAS Language Reference: Dictionary*.

- $SEVERITY indicates the most severe status of any step in your SAS program. To see the value of $SEVERITY, use the SHOW SYMBOL $SEVERITY DCL command. Table 2.1 on page 43 correlates the value of $SEVERITY with the severity level of the step in your program.

Table 2.1 Severity Levels for $SEVERITY

Value of $SEVERITY	Severity Level
0	WARNING
1	SUCCESS
2	ERROR
3	INFORMATIONAL
4	FATAL

- $STATUS also indicates the most severe status of any step in your SAS program. An OpenVMS severity level is associated with each value. You can check the severity level of $STATUS with the SHOW SYMBOL $STATUS DCL command to determine the final status of your SAS job. Table 2.2 on page 44 shows the severity levels that are associated with $STATUS under the given conditions.

Table 2.2 Severity Levels for $STATUS

Condition	Severity Level	Return Code Value
All steps terminated normally	SUCCESS	"%X1801A261"
SAS issued warning(s)	WARNING	"%X1801A3E0"
SAS issued error(s)	ERROR	"%X1801A44A"
User issued the ABORT statement	INFORMATIONAL	"%X1801A303"
User issued the ABORT RETURN statement	INFORMATIONAL	"%X1801A30B"
User issued the ABORT ABEND statement	FATAL	"%X1801A4F4"
SAS terminated abnormally	FATAL	"%X1801A4FC"
Core internal error	FATAL	"%X1801A4E4"
Host internal error	FATAL	"%X1801A4EC"

Customizing Your SAS Session

No matter which mode of operation you use for running SAS, you may want to customize certain aspects of the SAS System. For example, you may want to change the line size or page size for your output, or you may want to see performance statistics for your SAS programs.

Under OpenVMS, you can customize the SAS System for your session in the following ways:

- Specify SAS system options when you invoke SAS with the SAS command. This method is usually used for one-time overrides of the system option settings that would otherwise be in effect for your SAS session. See “Specifying System Options in the SAS Command” on page 45.
- Specify SAS system options in a configuration file. This method is useful if you, as an individual user, always want to override the values of system options that are specified in your site's system configuration file, or if you always want particular system options to be in effect for a particular job. See “Configuration Files” on page 46.
- Execute SAS statements (such as OPTIONS, LIBNAME, and FILENAME statements) in an autoexec file. This method is most useful for specifying options and files that pertain to a particular SAS application. See “Autoexec Files” on page 47.
- Execute an OPTIONS statement in a SAS program. See “OPTIONS Statement” on page 49.
- If you are using the SAS windowing environment, change SAS system option settings from within the System Options window. See “System Options Window” on page 50.
- If you are using the SAS windowing environment, specify a SASUSER library that contains a user profile catalog. See “SASUSER Library” on page 51.
- Define or redefine OpenVMS logical names that the SAS System uses. See “OpenVMS Logical Names That SAS Uses” on page 52.
- Specify SAS system options using the VMS_SAS_OPTIONS DCL symbol.

Note: For information about customizing your SAS windowing environment, see Chapter 4, "Customizing the SAS Windowing Environment," on page 71. △

Specifying System Options in the SAS Command

The way you specify the SAS command determines the mode of operation that you use for running SAS as well as the default SAS system options. The general form of the SAS command is

$ SAS8/*system-option-list filename*

Both *system-option-list* and *file-specification* are optional. You can include *system-option-list* for any mode of operation. If you include *file-specification*, then SAS is invoked in noninteractive mode. If you do not include *file-specification*, then the mode of operation will be the SAS windowing environment. For details about invoking SAS in the different modes of operation, see "Introduction" on page 18.

All SAS system options can be specified in the SAS command. Under OpenVMS, each option is preceded by a forward slash (/).

In the following example, the LINESIZE= system option tells SAS to use a line length of 80 characters for the log file, the procedure output file, and the print file:

```
$ SAS8/LINESIZE=80/PRINT=SYS$LOGIN:TEST.OUT
```

The PRINT= system option tells SAS to route the procedure output to the file SYS$LOGIN:TEST.OUT.

The next example invokes SAS in noninteractive mode, specifying the program file MYPROG and the LINESIZE= and PAGESIZE= system options:

```
$ SAS8/LINESIZE=60/PAGESIZE=80 MYPROG
```

As the examples show, system options that take a value (such as LINESIZE= and PRINT=) are specified in the following form:

```
/option-name=value
```

Note: Any option value that is entered on the OpenVMS command line within single quotation marks (') is resolved to its symbol value before it is processed by the SAS System. Any quoted value that should not be resolved as a symbol must be enclosed in double quotation marks ("). For example, the values for the system options FMTSEARCH=, INITSTMT=, and SYSPARM= must be enclosed in double quotation marks. △

Other system options can be thought of as on (enabled) or off (disabled). Specifying just the keyword enables the option; specifying the keyword with the prefix NO disables the option. In the following example, the CENTER and STIMER system options are disabled:

```
$ SAS8/NOCENTER/NOSTIMER
```

If no system options are specified in the SAS command, a configuration file, or an autoexec file, then the default system options that are shipped with the SAS System are in effect. However, your system manager may have overridden those default options. Ask your system manager for details about the default system options at your site.

For more information about SAS system options, see Chapter 18, "System Options," on page 387 and *SAS Language Reference: Dictionary*.

Configuration Files

A SAS *configuration file* contains SAS system options that are set automatically when you invoke SAS. Configuration files can contain only SAS system option settings and are processed *before* the SAS System initializes.

For Version 8, the configuration file is typically named SASV8.CFG. This file typically resides in your home directory.

Under OpenVMS, the OpenVMS logical name SAS$CONFIG is used to refer to SAS configuration files. This logical name can exist in one or more of the process-, job-, group-, or system-level logical name tables. Therefore, four types of configuration files can be created:

- The *process-level configuration file* should contain the system option settings that you, as an individual user, want to have in effect each time you invoke SAS.
- The *job-level configuration file* should contain the system option settings that you want to have in effect for a particular SAS job. (You can have multiple job-level configuration files, one for each job or for a group of jobs.)
- The *group-level configuration file* should contain the system option settings that your group manager or system manager has defined for members of your workgroup.
- The *system-level configuration file* should contain the system option settings that your system manager has defined for all users at your site.

Ask your system manager which of these configuration files are used at your site.

Creating a Configuration File

To create a configuration file, follow these steps:

1. Use any text editor to write SAS system options into an OpenVMS file. Use .CFG as the file type.
2. Specify one or more system options in each line. A configuration file can contain any system option except the CONFIG= and VERBOSE options. (If either of these options appears in a configuration file, it is ignored; no error or warning message appears.) Use the same syntax that you would use for specifying system options with the SAS command (see “Specifying System Options in the SAS Command” on page 45)—except don't include the SAS command itself. For example, a configuration file might contain the following lines:

   ```
   /SASUSER=DISK:[JQK.SASUSER]/WORK=[JQK.SASWORK]
   /DMS/LINESIZE=80/PAGESIZE=60
   /FULLSTIMER
   ```

   Note: You cannot include comment lines in a configuration file. △
3. Close the new configuration file.
4. Create the logical name SAS$CONFIG in the appropriate logical name table. For example, the following DEFINE DCL command creates the logical name SAS$CONFIG in the process-level logical name table.

   ```
   $ DEFINE SAS$CONFIG-
   _$ DISK:[DIRECTORY]MYCONFIG.CFG
   ```

   For more information about creating logical names, see *OpenVMS User's Manual*.

Specifying a User Configuration File

If you have created the OpenVMS logical name SAS$CONFIG, then SAS automatically executes the configuration file that is associated with that logical name. If SAS$CONFIG exists in more than one logical name table, then SAS executes the configuration files in the order in which they are listed in "Precedence for System Option Specifications" on page 50.

Alternatively, you can use the CONFIG= system option in the SAS command to tell SAS where to find your configuration file. For example, the following SAS command invokes SAS and tells it to use the process-level configuration file MYCONFIG.CFG:

```
$ SAS8/CONFIG=DISK:[DIRECTORY]MYCONFIG.CFG
```

Displaying the Contents of Configuration Files

When you invoke SAS, you can use the VERBOSE system option to write the contents of all configuration files to your OpenVMS display as the SAS System initializes.

For example, suppose your site has a system-level configuration file (defined by the logical name SAS$CONFIG in your system-level logical name table) that contains the following system options:

```
/LINESIZE=80/PAGESIZE=60
```

Suppose that you have also created your own configuration file, MYCONFIG.CFG, and that it contains the following options:

```
/FULLSTIMER
```

Now suppose you use the following command to invoke SAS:

```
$ SAS8/CONFIG=MYCONFIG.CFG/VERBOSE
```

The contents of both the system-level configuration file and MYCONFIG.CFG are written to your OpenVMS display, as follows:

```
The /VERBOSE option was specified.
SYSTEM SAS$CONFIG file
/LINESIZE=80/PAGESIZE=60

PROCESS SAS$CONFIG or /CONFIG= file
/FULLSTIMER
```

Autoexec Files

Unlike configuration files, which can contain SAS system options, an *autoexec file* can contain valid SAS statements. Autoexec files are processed immediately *after* the SAS System initializes but before it processes any source statements.

For Version 8, the autoexec file is named AUTOEXEC.SAS. This file typically resides in your home directory.

For example, an autoexec file could contain the following lines:

```
options fullstimer linesize=75;
libname mylib 'dev:[homedir.subdir]';
dm 'wait 0';
```

In this example, the OPTIONS statement sets some SAS system options, the LIBNAME statement assigns a libref, and the DM statement executes a SAS windowing environment command.

Note: Some SAS system options can be specified only when you invoke the SAS System. These system options cannot be specified in an OPTIONS statement; therefore, they cannot be specified in an autoexec file. "Summary of SAS System Options under OpenVMS" on page 387 tells where each SAS system option can be specified. △

Because autoexec files are processed after SAS is initialized, setting the NODATE and LINESIZE= options in a configuration file affects the appearance of the SAS log header, whereas setting NODATE and LINESIZE= in an autoexec file does not. An OPTIONS statement in an autoexec file is equivalent to submitting an OPTIONS statement as the first statement of your SAS session.

Under OpenVMS, the OpenVMS logical name SAS$INIT is used to refer to SAS autoexec files. This logical name can exist in one or more of the process-, job-, group-, or system-level logical name tables. Therefore, four types of autoexec files can be created:

- The *process-level autoexec file* should contain the SAS statements that you, as an individual user, want to execute immediately after your SAS session is initialized.
- The *job-level autoexec file* should contain the SAS statements that you want to execute for a particular SAS job.
- The *group-level autoexec file* should contain SAS statements that your group manager or system manager has specified for members of your workgroup.
- The *system-level autoexec file* should contain the SAS statements that your system manager has specified for all users at your site.

Ask your system manager which of these autoexec files are used at your site.

Creating an Autoexec File

To create an autoexec file, follow these steps:

1. Use any text editor to write SAS statements into an OpenVMS file. Use .SAS as the file type.
2. Type in the SAS statements that you want to include.
3. Close the new autoexec file.
4. Create the logical name SAS$INIT in the appropriate logical name table. For example, the following DCL DEFINE command creates the logical name SAS$INIT in the process-level logical name table.

```
$ DEFINE SAS$INIT-
_$ DISK:[DIRECTORY]MYEXEC.SAS
```

   For more information about creating logical names, see *OpenVMS User's Manual*.

Specifying an Autoexec File

If you have created the OpenVMS logical name SAS$INIT, then SAS automatically executes the statements in the autoexec file that is associated with that logical name. If SAS$INIT exists in more than one logical name table, then SAS executes the autoexec files in the order in which they are listed in "Precedence for System Option Specifications" on page 50.

Alternatively, you can use the AUTOEXEC= system option in the SAS command to tell SAS where to find your autoexec file. For example, the following SAS command invokes SAS and tells it to execute the autoexec file MYEXEC.SAS:

```
$ SAS8/AUTOEXEC=DISK:[DIRECTORY]MYEXEC.SAS
```

Displaying Autoexec Statements in the SAS Log

SAS statements that are submitted from an autoexec file usually are not displayed in the SAS log. However, if you specify the ECHOAUTO system option when you invoke

SAS, then SAS writes (or "echoes") the autoexec statements to the SAS log as they are executed. For example, suppose your autoexec file is MYEXEC.SAS and that it contains the following SAS statements:

```
options fullstimer linesize=75;
libname mylib 'dev:[homedir.subdir]';
dm 'wait 0';
```

If you use the following command to invoke SAS, then the contents of MYEXEC.SAS will be written to the SAS log.

```
$ SAS8/AUTOEXEC=MYEXEC.SAS8/ECHOAUTO
```

For more information about the ECHOAUTO system option, see *SAS Language Reference: Dictionary*.

OPTIONS Statement

You can use the OPTIONS statement to specify system option settings at any time during a SAS session, except within data lines or parmcard lines. Settings remain in effect for the duration of the session or until you reset them with another OPTIONS statement.

Not all system options can be specified in an OPTIONS statement. The summary table of system options, Table 18.1 on page 388, tells where each system option can be specified.

The following is an example of an OPTIONS statement:

```
options nodate linesize=72;
```

For more information about the OPTIONS statement, see *SAS Language Reference: Dictionary*.

Displaying System Option Settings

Most SAS system options are set to default values. To display the current settings of SAS system options, use either the OPTIONS procedure, the System Options window, or the GETOPTION function.

OPTIONS Procedure

The OPTIONS procedure writes to the SAS log all system options that are available under OpenVMS. Chapter 18, "System Options," on page 387 describes the system options listed by the OPTIONS procedure that are host-specific and that have host-specific behavior. *SAS Language Reference: Dictionary* describes all system options that are completely portable (that is, those that have no host-specific behavior) and that are portable but may be specified differently in various operating environments.

By default, the procedure lists one option per line with a brief explanation of what the option does. To list the options with no explanation, use the LIST option:

```
proc options list;
   run;
```

For more information about the OPTIONS procedure, see "OPTIONS" on page 344 and *SAS Language Reference: Dictionary*.

System Options Window

The SAS System Options window displays the settings of the SAS system options. The system options are grouped by their function within the SAS System. Each group has at least one subgroup.

To display the System Options window, do one of the following:

- Type **options** on the command line of any SAS windowing environment window or windowing procedure window and press RETURN.
- From the **Tools** menu, select **Options**, and then select **System...**.

You can select a group of system options by clicking on the icon to the left of the group name in the left side of the System Options window. To open a subgroup either click on the icon to the left of the subgroup name in left side of the window or double-click on the subgroup name in the right side of the window. You will see a list of the system options in that subgroup, along with their values and a brief description.

You can also use the System Options window to change the settings of system options for the duration of your SAS session. To change the setting of a system option either double-click on the name of the system option, or with cursor on the name of the system option press the right mouse button and select **Modify Value**. The Modify Value dialog box opens. You can then modify the setting of the system option as desired. Click on OK to save your changes. Click on Cancel to ignore any changes and close the Modify Value dialog box.

You can close the System Options window by doing one of the following:

- Double-click on the window menu button in the upper-left corner.
- Click on the window menu button in upper-left corner and select **Close** from the menu.

For help and additional information about the System Options window, click on Help in the window.

For additional information about system options settings, see "Summary of SAS System Options under OpenVMS" on page 387and *SAS Language Reference: Dictionary*.

GETOPTION Function

The GETOPTION function returns the value of a SAS system option or graphics option. It can be used within the DATA step or in conjunction with %SYSFUNC in open code. For additional details, see the GETOPTION function in *SAS Language Reference: Dictionary*.

Precedence for System Option Specifications

For many system options, different values may be specified in the SAS command, in a configuration file, in an OPTIONS statement (submitted in an autoexec file or in a SAS program), and in the System Options window. When the same system option is set in more than one place, the order of precedence is as follows:

1. System Options window or OPTIONS statement (submitted from a SAS session or job).
2. autoexec files that contain OPTIONS statements (after SAS is initialized but before the user supplies input):
   a. process-level autoexec file
   b. job-level autoexec file
   c. group-level autoexec file
   d. system-level autoexec file.

3 SAS command.

4 configuration files (as SAS is being initialized):

   a process-level configuration file
   b job-level configuration file
   c group-level configuration file
   d system-level configuration file.

5 VMS_SAS_OPTIONS DCL symbol:

   a local symbol definition
   b global symbol definition.

In other words, the System Options window or OPTIONS statement takes precedence over autoexec files; autoexec files take precedence over the SAS command; the SAS command takes precedence over configuration files; and the configuration files take precedence over the VMS_SAS_OPTIONS DCL symbol.

Precedence for Similar Types of Options

Some SAS system options have the same effect (and usually the same name) as other types of options. For example, the BUFSIZE= system option is analogous to the BUFSIZE= data set option. Also, under OpenVMS, the CC= system option is analogous to the CC= external I/O statement option that is described in "Host-Specific External I/O Statement Options" on page 361 in the FILENAME statement.

In the case of overlapping options, the SAS System uses the following rules of precedence:

- A value that is specified in a statement option (for example, an engine/host option in the LIBNAME statement or an external I/O statement option in the FILENAME, INFILE, or FILE statement) takes precedence over a value that is specified in a system option.
- A value that is specified in a data set option takes precedence over a value that is specified in a statement option.

SASUSER Library

The SAS System assigns a data library that has the libref SASUSER. The SASUSER library contains a SAS catalog that enables you to customize certain features of the SAS System while your SAS session is running and to save these changes. For example, in base SAS software, any saved changes that you make to function key settings or to window attributes are stored in a catalog named SASUSER.PROFILE. The SASUSER library can also contain personal catalogs for other SAS software products. You can also store SAS data files, SAS data views, SAS programs, and additional SAS catalogs in your SASUSER library.

In addition to storing function key settings and window attributes, the SASUSER.PROFILE catalog is used to store your DEFAULT.FORM. The DEFAULT.FORM is created by the FORM subsystem. It is used to control the default destination of all output that is generated by the PRINT command during a SAS windowing environment session. For information about the FORM subsystem, see "Host-Specific Frames of the FORM Window" on page 240 and *SAS Language Reference: Concepts*.

Under OpenVMS, the system-level logical name SAS$USER specifies the location of the default SASUSER data library. This logical name is defined when SAS is installed and points to SYS$LOGIN, your default login directory.

This name defines the location of the default SASUSER data library. For more information about how to change this library with the SASUSER= system option, see “SASUSER=” on page 435.

Creating Your Own SASUSER Libraries

By creating your own SASUSER libraries, you can customize the SAS System to meet the requirements of a number of different types of jobs. For example, suppose you want to create a user profile for a particular type of task that requires a unique set of key definitions.

To tell SAS which library to use as your SASUSER library, use the SASUSER= system option when you invoke SAS. For example, if you want to designate a directory named MYSUSER as your SASUSER library, you would use the following command:

```
$ SAS8/SASUSER=DISK:[MYSUSER]
```

Any profile changes that you make during your session are saved in the SAS catalog SASUSER.PROFILE, which is a file in the MYSUSER directory. These changes will be retained when you end your SAS session.

OpenVMS Logical Names That SAS Uses

A command procedure file that is loaded by the SAS installation procedure defines many logical names that affect certain aspects of SAS programs or SAS sessions. Most of these logical names are of interest only to system administrators. A command procedure file (SAS8_SYSTEM.COM) defines these same logical names at the system level. System administrators can see *Installation Instructions and System Manager's Guide: The SAS System under OpenVMS Alpha and VAX, Version 8* for details and can redefine these logical names if necessary.

Note: The installation procedure provides two command procedure files, SAS8.COM and SAS8_SYSTEM.COM. The SAS8.COM file defines process-level logical names. The SAS8_SYSTEM.COM file defines system-level logical names. △

You can also customize your SAS sessions by defining some of the logical names at the process or job level. (When a logical name is defined at both the system level and the job level, the job-level definition takes precedence.) In addition, some logical names that are *not* defined by the SAS8.COM or SAS8_SYSTEM.COM file at the system or process level can be defined by individual users at the process or job level (for example, SAS$CONFIG).

System-Level Logical Names That You Can Override

The following logical names are defined by the SAS8.COM or SAS8_SYSTEM.COM files during installation of the SAS System, but you can redefine them at the process or job level:

SASAUTOS
: defines to the SAS System the location of the autocall macro library, which contains SAS programs defining SAS macros. For more information, see “Autocall Libraries” on page 464.

SAS$EXTENSION
: defines to the SAS System the location of examples of user-written extensions to the SAS System, as well as the parts of the SAS System that are specific to the OpenVMS system. This is a rooted, concealed logical name.

SAS$GDEVICE
defines to the SAS System the output destination for a graphics device. SAS$GDEVICE defaults to your terminal, SAS$TERMINAL.

SAS$HELPLOC
defines the location of the online help files. SAS$HELPLOC corresponds to the HELPLOC system option. For more information, see "HELPLOC" on page 416.

SAS$LIBRARY
defines the search path that the SAS System uses to load executable images. SAS searches for executable images in the following order:

1. SYS$DISK:[ ]
2. SAS$ROOT:[PROCS]
3. SAS$ROOT:[IMAGE]
4. SAS$EXTENSION:[LOAD]

Note: If you define an OpenVMS search-list logical name to reference a list of directories, and then use the DCL command SET DEFAULT to change your default directory location to that logical name, you cannot invoke the SAS System. When you use the SET DEFAULT command, it modifies the SYS$DISK logical name to reference the new location. SYS$DISK is used in the definition of the SAS$LIBRARY logical name. The SAS System is unable to locate the files that it requires when SAS$LIBRARY and SYS$DISK are defined in this way. If you must set your default directory to an area that is defined by a search-list logical name, then remove references to SYS$DISK in all logical names that begin with SAS$. △

SAS$SECTION
defines the file specification for the native editor section file. For more information about native editors, see the command "TPU" on page 234.

SAS$TERMINAL
defines the output destination for a device. By default, SAS$TERMINAL is defined as SYS$OUTPUT.

SAS$USER
defines to the SAS System the location of the SASUSER data library. This logical name can be overridden by the SASUSER= system option. By default, SAS$USER is usually set to your default directory, SYS$LOGIN.

SAS$WORKROOT
defines the directory under which you want to create the WORK subdirectory. This can be overridden with the WORK= system option. By default, SAS$WORKROOT is defined as your current directory, SYS$DISK:[ ].

SAS$XDEFAULTS
defines the default location for X resource files that are used with Motif. For more information, see Chapter 4, "Customizing the SAS Windowing Environment," on page 71.

Other Logical Names That You Can Define

The following logical names are not defined by the SAS8.COM or SAS8_SYSTEM.COM file during installation, but you may want to define them at the process level as a way of customizing certain aspects of your SAS sessions:

SAS$ALTLOG
contains the name of the current alternate SAS log file, if one is created. SAS$ALTLOG corresponds to the ALTLOG= system option. For more information, see the system option "ALTLOG=" on page 397.

SAS$ALTPRINT
contains the name of the current alternate SAS list file, if one is created. SAS$ALTPRINT corresponds to the ALTPRINT= system option. For more information, see the system option “ALTPRINT=” on page 397.

SAS$APPLETLOC
defines the location of Java applets to the SAS System. For more information, see “APPLETLOC=” on page 398.

SAS$CONFIG
defines to the SAS System the location of a configuration file. For more information, see “Configuration Files” on page 46 and “Precedence for System Option Specifications” on page 50.

SAS$INIT
defines to the SAS System the location of an autoexec file. The AUTOEXEC= system option takes its value from SAS$INIT, if SAS$INIT is defined. For more information, see the system option “AUTOEXEC=” on page 399.

SAS$LOG
contains the name of the SAS log. SAS$LOG corresponds to the LOG= system option.

SAS$PRINT
contains the name of the SAS list file. SAS$PRINT corresponds to the PRINT= system option.

SAS$TRANTAB
specifies the names of translation tables that are used by various parts of the SAS System.

SAS$WORKLIB
points to your WORK subdirectory. For more information about the WORK subdirectory, see “The WORK Data Library under OpenVMS” on page 32.

SAS$X_STATUS
contains the OpenVMS status code that indicates whether an X command or X statement executed properly. The logical name is stored in the JOB logical name table and can be checked after the execution of any X command or statement. If the X command or statement was successful, the value is 1. Any other value indicates that the X command or statement was not successful.

The following program uses the GETLOG function to determine whether the X statement executed properly. If the X statement did execute properly, then the program continues; if it did not, then the program stops.

```
x 'create/dir [sasxyz.newdir]';
data _null_;
   x=getlog('sas$x_status');
   if x^="1" then do;
      put 'The directory was not created.';
      put 'The OpenVMS return code was:  ' x;
      put 'Program stopping.';
      abort return;
   end;
   else
      put 'Directory was created successfully.';
run;

libname mylib '[sasxyz.newdir]';
```

```
data mylib.test;
   input revenue expenses;
   profit=revenue-expenses;
   datalines;
39800.73 35678.93
28900.38 28456.99
40933.22 5683.33
;
```

Note: The ABORT RETURN statement not only stops the program, but also ends your SAS session. △

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Companion for the OpenVMS Environment, Version 8,* Cary, NC: SAS Institute Inc., 1999. 518 pp.

SAS® Companion for the OpenVMS Environment, Version 8

1–58025–526–4

1st printing, October 1999

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

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