CHAPTER
4

Using SAS Files

SAS Files and SAS Data Libraries

Under CMS, a *SAS file* is a specially structured CMS file. SAS files are logically grouped into *SAS data libraries*. A SAS data library can reside either on a CMS minidisk, in an *accessed* Shared File System (SFS) directory, or in an *unaccessed* SFS directory. An *accessed* SFS directory is one that you have identified to the operating environment with the CMS ACCESS command.

The logical grouping of a SAS data library is done according to filetype and filemode or directory. That is, files that have the same CMS filetype in a minidisk or in an SFS directory are members of the same SAS data library.

Note: The term *disk* is used to refer to both a minidisk and an SFS directory. △

Choosing between Disk and Tape Storage

Under CMS, permanent SAS files can be stored in the following formats:

- disk format on disk.
- sequential format on tape. Sequential format is commonly referred to as *tape format*.
- sequential (tape) format on disk.

Advantages of Tape Format

Tapes are able to accommodate large files for which disk storage is impractical. In addition, CMS Version 6 and Version 7 SAS files that are in tape format can be read by SAS under OS/390 and VSE (Version 8 tape format libraries cannot be read by SAS under VSE).

Disadvantages of Tape-Format Files on Tape

Tape-format files *on tape* have the following disadvantages:

- Accessing tapes is usually slower and more cumbersome than accessing files on disk because the tapes must be mounted by the operator.
- For unlabeled tapes, you must position the tape to the correct file unless you are accessing the first file on the tape.
- Only one tape file can be accessed at a time. This means that you cannot merge or concatenate two or more tape-format SAS data sets from the same tape. For example, this DATA step will not be executed:

```
data new;
   set tape.one
      tape.two;
run;
```

- If a SAS file on tape is modified, replaced, or deleted, any file that follows (whether it is another SAS file or some other type of file) cannot be accessed.
- You cannot use the EDITOR procedure for SAS data sets that are stored in tape format because PROC EDITOR uses direct-access methods as opposed to sequentially processing one observation after another. Likewise, you cannot use the SAS/FSP FSVIEW or FSEDIT procedures to process a tape-format data set.
- SAS data sets in tape format cannot be accessed randomly with the POINT= and NOBS= variables in the SET statement.
- There are no facilities to rename SAS files in tape format.
- Whether you are using tape or disk, each SAS file in tape format is stored as a single, physical file as long as each file has a unique libref.

  Note: If you write more than one SAS file in tape format using the same libref, they are written as one physical, sequential file. △

Disadvantages of Tape-Format Files on Disk

Tape-format files *on disk* have many of the same disadvantages as tape-format files on tape:

- You cannot use the EDITOR procedure for SAS data sets that are stored in tape format because PROC EDITOR uses direct-access methods as opposed to sequentially processing one observation after another. Likewise, you cannot use SAS/FSP FSVIEW or FSEDIT procedures to process a tape-format data set.
- SAS data sets in tape format cannot be accessed randomly with the POINT= and NOBS= variables in the SET statement.
- There are no facilities to rename SAS files in tape format.
- Whether you are using tape or disk, each SAS file in tape format is stored as a single, physical file as long as each file has a unique libref.

  Note: If you write more than one SAS file in tape format using the same libref, they are written as one physical, sequential file. △
- If you replace or delete an existing tape-format disk SAS file that is in a sequential file with other SAS files, any *SAS* file that follows in the sequential file can no longer be accessed.

CAUTION:

If you are using Version 5 sequential format, never use a libref or DDname that begins with TAPE for external files of any kind. △

See “Working with SAS Files on Tape” on page 36 for more information about tape handling under CMS.

Identifying SAS Data Libraries

Under other operating environments, a typical method of identifying a SAS data library (or an individual file in a SAS data library) is to first use the SAS LIBNAME statement or function within a SAS session or program to assign a libref to the data library. (See “LIBNAME” on page 243 for complete information about using the LIBNAME statement under CMS.) The libref identifies the library and some of its characteristics to the SAS System. Thereafter, you can use the libref as a convenient way of referring to the library in your SAS programs.

However, keep in mind the following points when you identify SAS data libraries under CMS:

- Under CMS, you often do not need to assign a libref to disk-format data libraries. (For sequential files on tape or on disk, a libref is always required. A libref is also required for disk-format files that are stored in an unaccessed SFS directory.) See “Working with SAS Files on Disk” on page 34 for an explanation of how SAS locates disk-format data libraries.
- Under CMS, you can also use the CMS FILEDEF command to assign a DDname to the data library. Thereafter, you can use the DDname just as you would use a libref. However, see “Advantages of Using the LIBNAME Statement or Function” on page 30 for some important considerations. For information about the CMS FILEDEF command, see “Using the CMS FILEDEF Command” on page 32 .
- In DMS mode, which is accessible through the SAS Explorer or the LIBASSIGN command, is the “New Library” dialog for assigning librefs.

Using the LIBNAME Statement or Function

Advantages of Using the LIBNAME Statement or Function

Although you can use the CMS FILEDEF statement to assign DDnames to your SAS data libraries, there are several reasons for using the LIBNAME statement or function (and librefs) instead:

- The CMS FILEDEF command is not portable to other operating environments. The LIBNAME statement or function is portable with minor changes to the *physical name* and options parameters.
- If you use the LIBNAME statement or function, you can allocate your data library for only as long as you need it, and then "free" (deallocate) it. By contrast, DDnames that are allocated externally remain allocated for the duration of the SAS session or job. (The LIBNAME CLEAR statement clears an externally allocated libref, but it does not deallocate the file. See “Clearing Librefs and DDnames” on page 46 .)
- DDnames that are allocated externally cannot be reassigned later by a LIBNAME statement or function. You receive an error message in the SAS log that states that the DDname is currently assigned.
- By using macro statements and the LIBNAME statement or function, you can conditionally allocate files.
- You cannot assign an engine when you allocate a file externally. SAS uses the procedure described in “How SAS Assigns an Engine When No Engine Is Specified” on page 45 to determine which engine to use. It is more efficient to specify an engine explicitly in a LIBNAME statement or function. Also, the following SAS engines must be specified in a LIBNAME statement or function because they are not assigned by default: XPORT, BMDP, SPSS, SPSSX, OSIRIS, V5TAPE, V6TAPE, V7TAPE, V8TAPE, and REMOTE.
- DDnames that are allocated externally are not included in the list that is produced by the LIBNAME LIST statement or in the LIBNAME window until after they

have been used as librefs in your SAS session. (See “Listing Your Current Librefs” on page 47 .)

LIBNAME Statement Syntax

This section provides an overview of the LIBNAME statement. For complete information about the LIBNAME statement see “LIBNAME” on page 243.

The general form of the LIBNAME statement is

LIBNAME *libref <engine> 'physical-name' <engine/host-options>*;

libref

is the logical name by which the library is referenced during your SAS session. The libref must begin with a letter and must contain one to eight characters consisting of letters or numbers.

When choosing a libref, follow the rules for SAS names, but do not use underscores. Also observe the restrictions listed in “Restrictions on Librefs” on page 32 .

To read, update, or create files that belong to a permanent SAS data library, you must include the libref as the first part of a two-level SAS member name in your program statements, as follows:*

```
libref.member
```

engine

tells SAS which engine to use for accessing the library. See Table 5.1 on page 54 for information about valid engine names. If you do not specify an engine, SAS uses the procedures described in “How SAS Assigns an Engine When No Engine Is Specified” on page 45 to assign an engine for you. If the engine name that you supply does not match the actual format or attributes of the data library, then any attempt to access the library will fail.

'physical-name'

enclosed in quotation marks, describes the physical location of the library. The *physical-name* can be specified in the following ways:

'filemode'

specifies the disk-mode letter or the disk-mode letter and optional filemode access number. Use this form for *physical-name* when you want to use a filemode other than the default. If you specify the filemode as a pair of single quotation marks (`''`) or as an asterisk surrounded by single quotation marks (`'*'`), SAS uses the standard CMS search order to locate an existing SAS library. If the library exists on more than one minidisk, then SAS stops searching as soon as it finds a member. Consequently, only the member on that minidisk is used. Otherwise, if the library does not exist, the assignment defaults to the first R/W accessed disk.

'filetype filemode'

specifies the filetype to be used for the library and the disk-mode letter. *filemode* can also include a filemode access number. Use this form for *physical-name* when you want to use the libref as an alias for the filetype.

* An exception is a SAS file in the WORK or USER library. In this case, you can use a one-level name. See “Directing Temporary SAS Data Sets to the USER Library” on page 23 for more information about the USER library.

'filetype sfs-dir'
: specifies an SFS directory to be used as a SAS library. Use this form for *physical-name* when you want to use the libref as an alias for the filetype.

'sfs-dir'
: specifies an SFS directory to be used as a SAS library.

'filename filetype filemode'
: specifies the complete CMS fileid. Use this form for *physical-name* with the BMDP, OSIRIS, SPSS, and XPORT engines.

('filetype-1 <filemode-1 | SFS-directory-1>' . . .'filetype-n <filemode-n | SFS-directory-n'>)
: specifies a concatenation of more than one library that will be accessed by SAS in order of specification using a single fileref. See "Concatenating SAS Data Libraries" on page 33 for more information on concatenation of SAS data libraries.

'TAP*n*'
: specifies the tape device for a sequential library. *n* is a hex character from 0 through F.

engine/host-options
: are options that apply to the SAS data library.

Note: The libref remains valid for the duration of the SAS job or session unless you clear it. See "Clearing Librefs and DDnames" on page 46 for information about clearing a libref. △

Restrictions on Librefs

Under CMS, you should observe the following restrictions on librefs:

- Do not use a libref that is reserved for use by the SAS System, as described in "CMS Filetypes Used by SAS" on page 9.
- Do not use **SAS** as a libref; it is reserved as the filetype for files that contain SAS programming statements in noninteractive SAS programs and in the %INCLUDE statement.
- Do not use as a libref a filetype that is reserved by CMS. (See the *VM/ESA CMS User's Guide* for a list of filetypes that are reserved by CMS.)
- Do not use the filetype of an external file as the libref of a SAS file.
- Librefs that begin with TAPE are reserved for SAS files that are written in the Version 5 tape format. Use a libref that begins with TAPE only for Version 5 tape format files.
- Specify the SAS system option NOREPLACE to prevent existing SAS data sets from being replaced.

Using the CMS FILEDEF Command

There are several advantages to using the LIBNAME statement or function to make your SAS data libraries available to your SAS programs. (See "Advantages of Using the LIBNAME Statement or Function" on page 30 .) However, in most cases you can also use the CMS FILEDEF command for this purpose.

Note: You cannot use the CMS FILEDEF command to assign a DDname to a SAS data library that resides on an unaccessed SFS directory. △

If you choose to issue a CMS FILEDEF command for a SAS data library, then use the following form of the command:

FILEDEF *DDname* DISK *dummy filemode*

DDname
: is the libref that you want to use for the data library.

dummy
: specifies any valid character string for the filename and filetype positions in the command. You can use the same value in both positions, and SAS will substitute the correct filename and filetype. Thus, you need to issue only one FILEDEF command per libref, regardless of the number of SAS data libraries that have or will have that libref.

filemode
: references the correct minidisk for the data library.

Remember, though, that if you choose to issue your own CMS FILEDEF command, you cannot subsequently issue a LIBNAME statement or function that uses the DDname that is assigned by the FILEDEF command as a libref.

Using a DDname as a Libref

After a DDname has been assigned, you can use it in a SAS job in the same way you would use a libref. For example:

```
proc contents data=books._all_;
run;
```

The first time that the DDname BOOKS is used in this manner, SAS assigns it as a libref for the SAS data library.

When a DDname is allocated externally, it is not listed by the LIBNAME LIST statement or in the LIBNAME window until after you have used it as a libref in your SAS session. (See “Listing Your Current Librefs” on page 47 .)

Concatenating SAS Data Libraries

The LIBNAME statement and the SASHELP=, USER=, and MAPS= system options can be specified with a concatenated series of multiple SAS data libraries. SAS accesses a concatenated library as an ordered series of individual libraries. The libraries in a concatenation can use different engines and can physically exist in any storage system (CMS, SFS directory, minidisk, MACLIB, etc.).

The syntax for specifying a concatenated SAS data library is as follows:

(’*filetype-1* <*filemode-1* | *SFS-directory-1*>’, . . .’filetype-n <*filemode-n* | *SFS-directory-n*>’)

Examples

```
libname mylib ('mylib a' 'mylib pool:.mysaslib');
```

```
libname twolibs ('first a' 'second a');
```

In the second example, the SAS data libraries FIRST and SECOND appear to SAS as a single library when the libref TWOLIBS is referenced. SAS searches for input members using the A disk, with the filetype FIRST searched first. If the member is not found in FIRST, SAS searches the filetype SECOND. New members receive the filetype FIRST on the A disk.

Working with SAS Files on Disk

Writing to SAS Files on Disk

To write a disk-format SAS file, you usually need only to specify the two-level filename in the appropriate SAS statement. That is, in most cases you do not need to assign a libref to the file because SAS automatically assigns librefs for permanent SAS files on disk. (However, see "When to Assign a Libref for a SAS File on Disk" on page 35 for exceptions and for performance considerations.) For example, suppose you specify the SAS filename MYLIB.TASTEST in a SAS statement. In CMS terms, this is equivalent to filename TASTEST and filetype MYLIB:

SAS name	CMS name
MYLIB.TASTEST	**TASTEST MYLIB**
libref.filename	filename filetype

Because the SAS filename includes no filemode or directory identifier, SAS uses the following procedure to determine where to write the file:

1. First SAS checks to see whether the libref MYLIB was previously assigned by a SAS LIBNAME statement or function and is still in effect. If so, SAS writes the new file to the minidisk or SFS directory that was indicated by the LIBNAME statement or function.
2. If MYLIB is not a current libref, then SAS checks to see whether MYLIB was previously assigned as a DDname by a CMS FILEDEF command and is still in effect. If so, SAS writes the new file to the minidisk or SFS directory that was indicated by the FILEDEF command.
3. If MYLIB is neither a current libref nor a current DDname, then SAS searches all accessed minidisks or SFS directories (in the standard search order) to see if there are any SAS files that have the filetype MYLIB. To do this, SAS must read each file to determine whether it is a SAS file. In some cases this can cause significant overhead, which can be avoided by using the LIBNAME statement or function. If SAS finds a SAS file with the filetype MYLIB, then it writes the new file to the same minidisk or SFS directory (unless the minidisk is accessed as READ-only, and then an error message is issued).
4. If no matching filetype is found, then SAS writes the file to the first R/W disk, using the libref MYLIB as the filetype.

Reading SAS Files on Disk

To read an existing SAS file that is in disk format, you usually need only to specify the two-level filename in the appropriate SAS statement. That is, in most cases, you do not need to assign a libref to the file. (However, see "When to Assign a Libref for a SAS File on Disk" on page 35 for exceptions and performance considerations.) For example, suppose you specify the SAS file MYLIB.TASTEST in a SAS statement. In CMS terms, this is equivalent to filename TASTEST and filetype MYLIB:

SAS name	CMS name
MYLIB.TASTEST	**TASTEST MYLIB**
libref.filename	filename filetype

Because the SAS filename includes no filemode or directory identifier, SAS uses the following procedure to locate the file:

1 First SAS checks to see whether the libref MYLIB was previously assigned by a SAS LIBNAME statement or function and is still in effect. If so, SAS refers to the minidisk or SFS directory that was indicated by the LIBNAME statement or function.

2 If MYLIB is not a current libref, then SAS checks to see whether MYLIB was previously assigned as a DDname by a CMS FILEDEF command and is still in effect. If so, SAS refers to the minidisk or SFS directory that was indicated by the FILEDEF command.

3 If MYLIB is neither a current libref nor a current DDname, then SAS searches all accessed minidisks or SFS directories (in the standard search order) to locate the file TASTEST MYLIB.

   Note: If the file exists on more than one minidisk or directory, SAS stops searching as soon as it finds the first TASTEST MYLIB file. If you have multiple files with the same name and filetype on different minidisks or directories, and you want to read a file that would not be located first according to the standard search order, then you must assign and use a libref. △

When to Assign a Libref for a SAS File on Disk

Under CMS, when you are reading from and writing to *disk*, you do not always have to assign a libref. (See "Reading SAS Files on Disk" on page 34and "Writing to SAS Files on Disk" on page 34 for explanations of why librefs generally are not required for disk-format files.) You may want to do so, though, for portability and clarity.

In the following examples, *filemode* can be specified in either of two ways:

□ a filemode letter that represents an accessed minidisk or an SFS directory.

□ an SFS directory name, which can be fully qualified (for example, FILEPOOL:USERID.DIR) or relative to the current user (for example, .DIR).

You should assign a libref under the following circumstances:

□ You want to read from or write to a SAS file that is not on your default disk. A libref is not strictly necessary in this case, but using one speeds processing, because SAS does not have to search multiple disks in order to locate the file. Use the following syntax:

  LIBNAME *libref 'physical-name'*;

  A physical name can be specified as either a filemode or an SFS directory. For example:

  LIBNAME MYLIB '.MYLIB';

□ You want to use a libref as an alias for the CMS filetype (valid only with the BASE engine). Use the following syntax:

  LIBNAME *libref 'filetype filemode'*;

  For example:

```
LIBNAME MYLIB 'REPORTS A';
```

- You want to speed processing by telling SAS which library engine to use. Use the following syntax:

  LIBNAME *libref engine 'physical-filename'*;

  For example:

  ```
  LIBNAME MYLIB V6 'REPORTS A';
  ```

Note: If you omit the *engine* argument, BASE is assumed by default. △

Working with SAS Files on Tape

When you write more than one SAS data library on a tape, remember that each library is a single tape file as long as the same libref is used for each SAS library member. A SAS data library that contains more than one member on tape is just one tape file.

Under CMS, SAS uses CMS tape-handling facilities to process tape libraries. Before you use tape libraries, see "Tape Processing" on page 36. Also, read about tape processing under CMS in the *CMS User's Guide* and in the *CMS Command Reference*.

Tape Processing

The SAS System under CMS enables you to exploit the tape-handling facilities that are provided by CMS. Both SAS data libraries and external files are accessed through the Basic Sequential Access Method (BSAM). This standard interface supports labeled or unlabeled tapes as well as multifile and multivolume facilities. Any supported external tape management system can be used. Read about tape processing in *VM/ESA CMS User's Guide* and in *VM/ESA CMS Command Reference*.

For Version 8, four tape engines are supported: V5TAPE (READ-only), V6TAPE, V7TAPE, and V8TAPE. Specifying only TAPE defaults to the current release.

Using the LIBNAME and FILENAME Options for Tape Processing

SAS uses a LIBNAME statement or function to access SAS data libraries and a FILENAME statement to access external files. See "Statements in the CMS Environment" on page 219 for details about these statements. The LIBNAME and FILENAME statements provide the following special options particular to tape processing:

DISP=MOD
: specifies that the COPY procedure is to append new members to the end of a sequential library that is allocated with this option. By default, the COPY procedure replaces all existing members in the sequential library with the new members copied from the source library. The DISP=MOD option is valid only for libraries that use the V6TAPE or later engine, and is valid only with the COPY procedure. When this option is in effect, SAS does not check the destination sequential library for duplicate members. Any duplicate members that are not renamed will not be accessed by SAS, since SAS will always access the existing member with the same name first.

LABEL=BLP *n*
: label processing is bypassed. An integer *n* (the default value is 1) causes positioning to a specific file on a multifile tape when it is opened.

LABEL=LABOFF
: indicates that no tape label processing is desired. A tape is not repositioned before open processing or after close processing; a CMS TAPE command is used to reposition the tape if needed.

LABEL=NL *n*
: unlabeled tapes are desired. An NL tape is always rewound at open to perform label checks. A standard labeled tape is not opened when NL is specified. After label checking, the tape is positioned to logical file *n*.

LABEL=SL *n*
: indicates that IBM standard labels are needed. Standard labeled tapes enable SAS data libraries or external files to span multiple tape volumes. If a tape management system is installed, the mounting of SL tapes can be deferred until the files are opened.

LEAVE=YES
: indicates that a multifile tape is not repositioned at open for LABOFF or BLP processing. For SL tapes, LEAVE=YES does not reposition before label processing. Omitting LEAVE or specifying LEAVE=NO causes a tape to be rewound and repositioned each time a file is opened.

SYSPARM=*value*
: is used to pass tape mounting parameters to an external tape management system. When *value* contains blanks or parentheses, use the form SYSPARM=? to cause an ENTER SYSPARM prompt. At the prompt, 130 characters of SYSPARM values can be entered. See the documentation for your external tape management system for valid values.

VOLID=*vvvvvv*
: defines a one- to six-character volume serial identifier for standard labeled tapes. When you use the VOLID= option, the VOL1 label on the tape is verified when a tape file is opened. When you omit VOLID=, no VOL1 check is made.

Using CMS Command Options for Tape Processing

When SL tapes are used, use a CMS LABELDEF command to supplement a SAS LIBNAME or FILENAME statement. The LABELDEF command enables you to define fields in the standard HDR1 or EOF1 tape labels. The defined fields or their default values are checked for input files or are written for output files. The LABELDEF command is required when multivolume tape files are used, when input checking is needed, or when specific values are to be written into standard labels. See the *VM/ESA CMS Command Reference* for more details. The filename field in a LABELDEF command should be the same name that is used as a libref in a LIBNAME statement or function or as the fileref in a FILENAME statement.

Mounting Tapes

The method that is used to have a tape mounted and a drive attached to your userid is specific to your computing installation. Some installations have special commands for tape mounts; others require that you send messages to your computer operator. Ask your SAS Installation Representative or other installation personnel about your local procedures.

If an external tape management system is available at your installation, it may be possible to defer mounting of tapes until the tape file is opened. In most cases, however, the tape drive must be attached to your userid with a tape mounted and ready before the tape file can be opened.

Positioning Tapes

When LABEL=SL, NL, or BLP, SAS automatically positions a tape to the proper file when the file is opened. No manual repositioning is needed in most cases.

When either LABEL=LABOFF or LEAVE=YES is used, tape positioning becomes a user responsibility. The CMS commands TAPE REW (rewind), TAPE FSF (forward space file), and TAPE BSF (backspace file) are needed to position a tape to the proper file before processing.

Miscellaneous Tape Notes

- Using DISP=MOD in a FILEDEF command or in a FILE, FILENAME, or INFILE statement to append output to the end of the file works only for SL tapes. SAS facilities that append data to existing tape data sets are also restricted to standard labeled tapes.
- Using DISP=MOD in a LIBNAME statement to specify that copied members are to be appended to the end of the library works only on sequential libraries that use the V6TAPE or later engines, and only in conjunction with the COPY procedure. Note that when DISP=MOD is asserted in a LIBNAME statement of a V6TAPE or later sequential library, SAS does not check the library for duplicate members.
- The SAS LIBNAME and FILENAME statements do not support the ALT= option that the FILEDEF command supports. If the ALT= option is needed for multivolume processing, use a CMS FILEDEF command instead of a LIBNAME or FILENAME statement.
- The SAS system option TAPECLOSE specifies tape positioning when a SAS data library on tape is closed. The REWIND and REREAD arguments are treated as REWIND. No processing occurs under CMS for the FREE and DISP arguments.
- If a tape is initialized with a TAPE WVOL1 command, it contains a dummy HDR1 record. To use this tape with a LIBNAME statement or function, issue a LABELDEF command with a file identifier that is specified to cause the dummy HDR1 record to be rewritten.

Tape Processing Example 1: Creating a SAS Data Library without Label Processing

```
/* Assume that the tape has been premounted */
/* at virtual address 182                   */

libname favorite tape 'tap2';
cms tape rew;
data favorite.fruits( filedisp=new );
   set mylib.oranges;
run;
```

The TAPE REW command ensures that the tape is rewound to file 1. FILEDISP=NEW is needed to create the first member in any tape library.

Tape Processing Example 2: Adding a New Member to an Existing SAS Data Library

```
/* Assume a premounted tape at address 183 */

libname fall89 tape 'tap3' label=NL 2;
data fall89.scores;
   input student $11. test 3.0;
cards;
...more data lines...

run;
```

For NL tapes, positioning is handled by the SAS System under CMS. No TAPE commands are used. FILEDISP=NEW is not specified because the SAS data library already exists in file 2.

Tape Processing Example 3: Creating a Multivolume External File

```
/* Assume the first tape volume VM0202 is */
/* mounted at address 181                 */

cms labeldef test1 fid ? volid ?;
DMSLBD220R Enter dataset name:
external.test.file
DMSLBD441R Enter VOLID information:
vm0202 vm0203
DMSLBD441R Enter VOLID information:
 ...null line entered...

filename test1 tape 'tap1' label=sl;
data; file test1;
   do i = 1 to 100000;
      put i;
   end;
run;
DMSTLM428I TAP1(181) EOV1 label written on VM0202
DMSTVS265I Attempting to change tape volume for DDNAME TEST1
DMSTVS266I To cancel the tape volume switch, type CANCEL
DMSTVS268I Message sent to userid OPERATOR:
DMSTVS269I Mount tape volume VM0203 on virtual 181 with a write ring;
request number 1
```

The CMS interface module uses the arguments for the LABELDEF command to switch to the second volume when the first volume is filled and its EOV1 label has been written.

Tape Processing Example 4: Retrieving a Multivolume Tape File Managed by an External Tape Management System

```
/* Assume the tape management system */
/* supports deferred tape mounts     */
```

```
cms labeldef test2 fid ?;
DMSLBD220R Enter dataset name:
wx.test.file
filename test2 tape 'tap1' label=sl sysparm=queue;
Beginning DMSTVI SYSPARM processing.

data;
   infile test2;
   input x y z;
run;

Beginning DMSTVI OPEN processing
TAPE nnnn ATTACHED TO userid 0181

...more system output...

DMSTLM427I TAP1(181) EOV1 label read
Beginning DMSTVI EOV processing.
TAPE 0181 DETACHED
TAPE nnnn ATTACHED TO userid 0181

...more system output...

Beginning DMSTVI CLOSE processing.
```

The external tape system handles the tape mounts when the tape file is opened and when the volume switch occurs. The LABELDEF command identifies the file by its data set name.

Tape Processing Example 5: Retrieving a SAS Data Library from One Tape File and an External File from a Second Tape File

```
/* Assume the tape has been mounted as 184*/

libname first tape 'tap4' label=blp;
cms tape rew;
data a;
   set first.member;
run;

filename second tape 'tap4' label=blp 2 leave=yes;

data;
   infile second;
   input a b;
run;
```

The CMS TAPE REW command positions the tape initially to the load point. Because LEAVE=YES is specified, the tape is not repositioned when the file SECOND is read. Without LEAVE=YES, the program still runs, but when SECOND is opened, the tape is rewound and is spaced forward unnecessarily.

Assigning a Libref to a Sequential Library

When a LIBNAME statement or function defines a libref for a library that is in sequential format, SAS automatically issues a FILEDEF command for the library. If you issue a FILEDEF command prior to the LIBNAME statement or function, then that FILEDEF is used.

Most of the options that you can specify in a CMS FILEDEF command can also be specified as LIBNAME statement or function options. Options that are needed for sequential processing are also available with the LIBNAME statement or function. (See "LIBNAME" on page 243 for a complete list of LIBNAME statement or function options.) Therefore, it is feasible for you to replace all FILEDEF commands in an existing SAS application with corresponding LIBNAME statement or functions.

To process a sequential data library, use the following form of the LIBNAME statement or function:

LIBNAME *libref* TAPE TAP*n*;

The *libref* is any valid SAS libref. TAPE designates the TAPE (sequential) engine, and TAP*n* specifies the tape device. Here are some examples of LIBNAME statements and explanations of where the library is assumed to be located:

- This example is a sequential library on the first R/W disk. It is assumed to be on disk because no tape device is specified.

  ```
  libname mytaplib tape;
  ```

- This example is a sequential library on tape device 181:

  ```
  libname mytaplib tape 'tap1';
  ```

Specifying an Old or New Library with the FILEDISP= Option

When writing a SAS file to a sequential library, you must be aware of the effects of the SAS data set option FILEDISP= . The FILEDISP= data set option tells SAS whether the file being written is a member of an existing SAS data library on the tape or is the first member of a new library.

When FILEDISP=OLD (the default) is in effect, SAS assumes that you are writing a member to an existing sequential library. During such a write, SAS searches the sequential library, which consists of a single SAS file, for a member that has the same name. If a matching member is found, SAS writes the new member over the existing member. Any members that appeared after the updated member are lost. If SAS does not find a matching member, it assumes that it has come to the end of the library. SAS then writes the new member at the end of the library.

If you specify FILEDISP=NEW, SAS assumes that you are creating a new library, and writes the new member at the beginning of the SAS file that represents the sequential library. After the initial write, SAS assumes FILEDISP=OLD for that libref. Subsequent writes can be submitted with FILEDISP=NEW, but the NEW value will be ignored. This prevents unintentional data loss after the initial write. To reiterate, FILEDISP=NEW affects only the first write to a given libref. For all subsequent writes to that libref, FILEDISP=NEW is ignored, and FILEDISP=OLD is assumed.

CAUTION:

Do not specify FILEDISP=NEW for the first write to a newly allocated libref that represents an existing sequential library. Doing so writes the new member at the beginning of the library and any other members in the library are lost. △

Remember, if you assigned the libref with a LIBNAME statement or function, you can reassign it with another LIBNAME statement or function. However, if you assigned the libref with a CMS FILEDEF command, you must use another FILEDEF command to change it.

Note: The COPY procedure ignores FILEDISP= and CMS FILEDEF commands when a sequential library is specified as the destination of the copy. By default, the COPY procedure removes all members in the destination sequential library and writes new members into the beginning of the SAS file. To preserve existing members in a V6TAPE or later library, that library must be allocated using the host option DISP=MOD in the LIBNAME statement. For details, see "Using the LIBNAME and FILENAME Options for Tape Processing" on page 36. △

Examples

- In the following SAS program, a SAS data file is written on an unlabeled tape that already contains one member of the same SAS data library. Assume that the tape has already been mounted, but not rewound and positioned. FILEDISP=OLD, the default, is in effect, so a TAPE REW command is issued to position the tape before the first member in the library. The DATA step writes the SAS data file on the tape after the existing SAS file.

```
cms tape rew;
libname favorite tape 'tap1';
data favorite.fruits;
   set mylib.oranges;
run;
```

- In the next example, an unlabeled tape already contains two external files. FILEDISP=NEW is specified for the SAS data set, so the tape must be spaced forward. Otherwise, SAS writes over the existing files.

```
cms tape rew;
cms tape fsf 2;
libname fall89 tape 'tap1';
data fall89.scores (filedisp=new);
   input student $11. test1 3.0 test2 3.0
      test3 3.0 test4 3.0;
   cards;
...more data lines...
;
```

- In the third example, a SAS data library is created on a new tape that will have standard labels. The TAPE WVOL1 command writes the volume label at the beginning of the tape, and the LIBNAME statement includes LABEL=SL. FILEDISP=NEW is specified because a new library is created.

```
cms tape rew;
cms tape modeset (den 6250;
cms tape wvol1 vm6111;
libname bank tape 'tap1' label=sl 1;
data bank.ncyield (filedisp=new);
   set agri.ncyield;
run;
```

- In the next example, a SAS data file is read from a tape with standard labels. The SAS data file is in the second file of the tape.

```
libname tapefile tape 'tap2' label=sl 2;
proc print data=tapefile.monitor;
   var weight age initbac baccnt chckdate;
```

```
      id subject;
      by dose;
   run;
```

- This next example illustrates the effects of specifying FILEDISP=NEW in an output step other than the first output step that follows the allocation of the libref. The library MYLIB is a tape format on disk library that has five members named ONE, TWO, THREE, FOUR, and FIVE. The first data step adds a new member, SIX, after member FIVE. The second data step writes over member FOUR, and members FIVE and SIX are lost. The value of FILEDISP=NEW is ignored and the default FILEDISP=OLD is assumed.

```
   libname mylib tape 'a';
   data mylib.six;
   f=1;
   output;
   run;
   data mylib.four (filedisp=new);
   d=1;
   output;
   run;
```

Working with SAS Files in Tape Format on Disk

In the CMS environment, SAS libraries can be stored on disk in tape format. Keep the limitations of sequential files in mind when choosing between tape format and disk format. Tape format on disk is not recommended because of the loss of functionality. For example, because the files in the SAS library cannot be randomly accessed, SAS data sets cannot be indexed nor can catalogs be used. Despite the limitation, tape-format SAS libraries can be processed by SAS under OS/390 and VSE.

As with SAS files on tape, each tape-format SAS file on disk is stored as a single file, as long as each SAS file has a unique libref. If you write multiple SAS files on disk in tape format using the same libref, they are all written to one sequential file, making access to a particular SAS file difficult.

To force tape format for a SAS library on disk, specify the sequential engine in the LIBNAME statement. The form of the LIBNAME statement to use is

LIBNAME *libref* TAPE <'*physical-name*'>;

Examples

- The following example causes the library to be written to the first R/W disk, using the default filename SASTAPE:

```
   libname raisins tape;
```

- Be sure to include *physical-name* if you want the file to be on a minidisk other than the default. For example, if you want the file to be written on your M-disk, use this LIBNAME statement:

```
   libname raisins tape 'm';
```

- The following example writes two tape-format SAS data files in separate libraries on the A-disk:

```
libname day1 tape 'a';
data day1.earn;
   input branch $ 1-20
      dept 22-24 @26 revenue 10.;
   cards;
...more data lines...
;

libname day2 tape 'a';

data day2.earn;
   input branch $ 1-20
      dept 22-24 @26 revenue 10.;
   cards;
...more data lines...
;
```

□ The next example reads the tape-format SAS data sets that were created in the previous example.

```
proc print data=day1.earn;
   id branch;
   var revenue dept;
run;
proc print data=day2.earn;
   id branch;
   var revenue dept;
run;
```

Accessing SAS Libraries in Segments

For improved performance you can create and read SAS data libraries that are stored in CMS segments. To implement SAS libraries in segments, install SAS accordingly, as specified in the installation instructions for the SAS System under CMS.

Access to a SAS library in a segment (READ-only) requires a slight modification of the usual LIBNAME statement, using the following syntax:

LIBNAME *libref* <*engine*<'*segment-name*'> **SEGMENT=YES**>;

libref
: is a name that identifies the library. If '*segment-name*' is not specified, then SAS uses the libref as the segment name, which means that it cannot contain characters such as an underscore (-), which is valid in a filetype but invalid in a segment name.

engine
: identifies the engine to be loaded to process the library. Only a base engine may be specified. Valid base engine names are V8, V7, or V6.

'*segment-name*'
: is a quoted string that provides the name of the segment if it is different from the libref. The string must consist of a single alphanumeric name of up to 8 characters from the set: a through z, A through Z, 0 through 9, @, #, $. Specify a segment name when you want to use the libref as an alias for the segment name.

SAS issues a SEGMENT RESERVE command for the segment and then loads the segment with a DIAGNOSE instruction. If either command fails for any reason, the LIBNAME statement fails.

A libref that is assigned to a library in a segment receives a path name of *segment-name* SG. The filetype is *segment-name* and the file mode is SG.

Examples

A segment named LABDATA contains a SAS library. The following example loads the library and assigns the libref LABDATA:

```
libname labdata segment=yes;
```

This next example loads a SAS library in the segment named SALEDATA and assigns the libref BOOKDATA:

```
libname bookdatab 'saledata' segment=yes;
```

Transporting SAS Files between Operating Environments

SAS supports three ways of transporting files between CMS and other operating environments: the XPORT engine, the CPORT and CIMPORT procedures, and the separately licensed SAS/CONNECT software.

The process of moving a SAS file to or from CMS with the XPORT engine or with the CPORT and CIMPORT procedures involves three general steps:

1. Convert the SAS file to the intermediate form known as *transport format*.
2. Physically move the transport format file to the other operating environment.
3. Convert the transport format file into a normal, fully functional SAS file, in the format that is required by the other operating environment.

For further information on the XPORT engine and about the CPORT and CIMPORT procedures, including limited restrictions, refer to *Moving and Accessing SAS Files across Operating Environments*.

SAS/CONNECT software enables you to move files between operating environments without using the intermediate transport format. For further information about SAS/CONNECT, including limited restrictions, refer to *SAS/CONNECT User's Guide*.

How SAS Assigns an Engine When No Engine Is Specified

In some cases, you may choose not to specify an engine name in the LIBNAME statement or function for a data library. For these situations, you need to know how SAS assigns an engine to the library.

- If you specify TAP*n* as the physical name, SAS assigns the engine that is specified by the SEQENGINE= system option. This is true for both new and existing libraries.
- For existing libraries on disk, SAS examines the library header record and assigns an engine that is based on its content. The engines follow:
  - V8 for a Version 8 standard library.
  - V8TAPE for a Version 8 sequential library.

- V7 for a Version 7 standard library.
- V7TAPE for a Version 7 sequential library.
- V6 for a Version 6 standard library.
- V6TAPE for a Version 6 sequential library.
- V5 for any Version 5 library.

Assigning Multiple Librefs to a Single SAS Data Library

You can assign more than one libref to the same SAS data library. Any assigned libref may be used to access the data library. In fact, you can use the librefs interchangeably.

For example, suppose that in two different programs you used different librefs for the same SAS data library. Later you develop a new program from parts of the two old programs, or you include two different programs with the %INCLUDE statement. In the new program, you could simply assign the two original librefs to the library and proceed.

Clearing Librefs and DDnames

To clear a libref that was assigned in a LIBNAME statement, issue a LIBNAME statement in the following form:

LIBNAME *libref* <CLEAR>;

Or issue a LIBNAME function:

LIBNAME (*libref*, CLEAR);

Alternatively, in the windowing environment, use the EXPLORER window to clear a libref. In the EXPLORER window (available as an option of the VIEW pmenu), select the libref, press the left mouse button (or the 3270 equivalent if you do not have a mouse), and select DELETE from the pull-down menu.

The LIBNAME statement, LIBNAME function, and the EXPLORER window deassign the libref and deallocate the library only if no other librefs are assigned to that library.

Note: Librefs are cleared automatically at the end of your SAS session. △

To clear a DDname that you have used as a libref in a SAS program, first clear the libref as shown. This step is necessary because the first time you use a DDname in a SAS program, SAS assigns it as a libref for the SAS data library. Then issue the CMS FILEDEF command with the CLEAR option to clear the DDname.

For example, suppose that you had used the CMS FILEDEF command to assign the DDname MYLIB to a SAS data library and that you subsequently used the DDname as a libref in a SAS program. The following two statements would clear both the libref and the DDname:

```
libname mylib clear;
x filedef mylib clear;
```

If the data library is currently being used by a DATA step or PROC, the LIBNAME statement or function fails.

Listing Your Current Librefs

You can use either the LIBNAME command or a form of the LIBNAME statement to list your current librefs. In both cases, DDnames for externally allocated data libraries are also listed, but only after you have used them as librefs in your SAS session. (See "Using a DDname as a Libref" on page 33 .)

- □ When you issue the LIBNAME command from a SAS window, the ACTIVE LIBRARIES window is displayed.

  The LIBNAME window lists all the librefs that are currently assigned for your session. The LIBNAME window lists the full physical path name of the SAS data library, as well as the engine that is used to access the data library.

  It lists all SAS files that are associated with the selected libref.

- □ The following form of the LIBNAME statement writes to the SAS log the attributes of all the librefs that are currently assigned for your session:

  ```
  LIBNAME _ALL_ LIST;
  ```

Managing Your CMS SAS Files and Libraries

SAS provides the CONTENTS, COPY, and DATASETS procedures to facilitate the management of SAS data libraries and files. Host-specific aspects of these procedures are described in "Procedures in the CMS Environment" on page 183. For complete discussions of these and other SAS utility procedures, see *SAS Procedures Guide*. The SAS procedures are described briefly in this section and are compared to the CMS commands that perform similar functions.

Listing SAS Files

Both the CONTENTS procedure and the CONTENTS statement in the DATASETS procedure can list all SAS files in a SAS library. The CONTENTS procedure can also retrieve the descriptor information at the beginning of a SAS data set.

To compare the file information that is generated by the CMS FILELIST command with the information that is generated by PROC CONTENTS, see Output 4.1 on page 47 and Output 4.2 on page 48. Output 4.1 on page 34 illustrates the file information that is displayed by issuing the following CMS FILELIST command

```
filelist * mylib a
```

This command, issued in CMS subset mode, requests a listing of all files on the A disk that have the filetype MYLIB.

Output 4.1 Sample Output from the CMS FILELIST Command

```
 USER1   FILELIST A0  V 108 Trunc=108 Size=3 Line=1 Col=1 Alt=0
Cmd  Filename Filetype Fm Format Lrecl Records Blocks   Date   Time
     3RANCH   MYLIB    A1 F       1024       3      1 06/01/99  8:39:55
     0HSCREEN MYLIB    A1 F       1024      13      3 06/01/99 16:47:25
     HOUSES   MYLIB    A1 F       1024       3      1 06/01/99 16:47:20


1=Help      2=Refresh 3=Quit   4=Sort(type)  5=Sort(date)  6=Sort(size)
7= Backward 8=Forward 9=FL /n 10=           11=XEDIT/LIST 12=Cursor

====>                                               X E D I T  1 File
```

Output 4.2 on page 48 shows the directory information that is retrieved by PROC CONTENTS for the same group of files. The following SAS statement lists all data sets in the SAS data library MYLIB. NODS specifies that no data set descriptor information be printed. The DIRECTORY option is assumed by default.

```
proc contents data=mylib._all_ nods;
```

Output 4.2 Sample Output from the CONTENTS Procedure

```
                                The SAS System                        3
                                            08:31 Monday, June 1, 1999

                              CONTENTS PROCEDURE

                             -----Directory-----

                          Libref:        MYLIB
                          Engine:        V8
                          Physical Name: MYLIB    A1

                         #  Name     Memtype  File  Last Modified
                                              Size
                         ----------------------------------------

                         1  HOUSES   DATA      9     01Jun1999:16:47:20
                         2  HSCREEN  CATALOG   9     01Jun1999:16:47:25
                         3  RANCH    VIEW      9     01Jun1999:08:39:55
```

Compare the CMS filenames that are listed by the CMS FILELIST command with the SAS filenames that are listed by the CONTENTS procedure. For example, look at the information for SAS file RANCH in Output 4.2 on page 48 . You see that its memtype is VIEW. Now look at the information in Output 4.1 on page 47 . You see an entry for a CMS file with a filename of 3RANCH. CMS recognizes the SAS file RANCH as 3RANCH because SAS added the prefix 3 to the filename so that it could recognize the file's SAS filetype. (See "SAS Filename Restrictions" on page 8 for more information about prefix characters.)

When you use SAS utilities to manage your SAS files, you do not need to be concerned about the prefix. But when you use CMS commands, you must use the prefix.

Copying SAS Files

The best ways to copy SAS files are with the SAS COPY procedure or with the COPY statement of the DATASETS procedure. However, sometimes it is possible to use CMS commands. CMS commands cannot convert SAS files to and from tape format; they cannot read DDnames that begin with TAPE. But you can use CMS commands to copy SAS files in the following ways:

- The COPYFILE command can copy SAS files from minidisk to minidisk.
- The MOVEFILE command can copy SAS files from minidisk to minidisk, from tape to tape, from minidisk to tape, or from tape to minidisk. If you move a disk-format SAS file to tape with the MOVEFILE command, the SAS file cannot be accessed until it is moved back to a minidisk with the MOVEFILE command.
- The DISK DUMP command punches a SAS file to the virtual card punch. The DISK LOAD command restores a file that was punched with the DISK DUMP command to its original form.
- The TAPE or VMFPLC2 DUMP command dumps disk files to tape. The TAPE or VMFPLC2 LOAD command restores disk files that were written to tape by the TAPE or VMFPLC2 DUMP commands. If you dump any SAS file to tape with the TAPE or VMFPLC2 DUMP commands, the SAS file cannot be accessed until it is restored to a minidisk with the TAPE or VMFPLC2 LOAD commands.

If you use a CMS command, remember to include any prefix characters in the CMS filename. (See "SAS Filename Restrictions" on page 8 for information about prefix characters.) For example, to copy the SAS data view PROJCTS MONTHLY into another SAS data view using the CMS COPYFILE command, you can issue the following command:

```
copy 3projcts monthly a 3rhouse monthly b
```

When you use the CMS COPYFILE command to copy a SAS data set that has an index file associated with it, be sure to copy the index file as well.

Deleting SAS Files

The SAS DATASETS procedure deletes and renames disk-format SAS files. You can also use the CMS ERASE command to delete some or all members of a SAS data library. For example, suppose you create a catalog called MYLIB.DATASCR (CMS file 0DATASCR MYLIB) that contains FSEDIT screens as entries. In a PROC FSEDIT statement you specify the SAS name for the file, which is the CMS filename without the prefix 0. For example, you can issue the following statement:

```
proc fsedit data=mylib.houses screen=mylib.hscreen;
```

But if you want to use the CMS ERASE command, specify the CMS filename 0HSCREEN, as in the following example:

```
ERASE 0HSCREEN MYLIB A
```

(See "SAS Filename Restrictions" on page 8 for more information about prefix characters.)

Renaming SAS Files

You can rename a disk-format SAS file with the CHANGE, EXCHANGE, and AGE statements in the DATASETS procedure or with the CMS RENAME command. If you use the RENAME command, remember to include any prefix characters in the filename.

Estimating the Size of a SAS Data Set

To obtain a rough estimate of how much space you need for a disk-format SAS data set that was created by a V6, V7, or V8 engine, follow these steps:

1 Use PROC CONTENTS to determine the size of each observation. (See "Using the CONTENTS Procedure to Determine Observation Length" on page 50.)
2 Multiply the size of each observation by the number of observations.
3 For V6 engines, add 10% for overhead. For V7 and V8 engines, add 1K (or 1,024 bytes) for overhead.
4 The result of this calculation will be in bytes. You can convert this value to blocks based on the capacity of your data storage device.

Note: This procedure is valid only for *uncompressed* native SAS data files that were created with a V6, V7, or V8 engine. △

Here is an example of a calculation for a V6 system that assumes a 2048-byte page size.

1 PROC CONTENTS reveals an observation size of 484 blocks.
2 484 blocks-per-observation * 16 observations = 7744 bytes. Round up 7744 to the next multiple of 2048, which is 10240 bytes.
3 (10240 * 1.10)+ 10240 = 11264 bytes.
4 11264/4096 = 2.75, which rounds up to 3 blocks

Here is an example of a calculation for a V8 system.
This example assumes an 8192-byte page size.

1 PROC CONTENTS reveals an observation size of 484 blocks.
2 484 blocks-per-observation * 16 observations = 7744 bytes. Round up 7744 to the next multiple of 8192, which is 16384 bytes.
3 16384 + 1024 = 17408 bytes.
4 17408/4096 = 4.25 , which rounds up to 5 blocks.

Using the CONTENTS Procedure to Determine Observation Length

To determine the length of each observation in a Version 8 SAS data set, create a SAS data set that contains one observation. Then run the CONTENTS procedure to determine the observation length. The CONTENTS procedure displays engine and host-dependent information, including page size, as well as the number of observations per page for uncompressed SAS data sets. For example, the following input produces a SAS data set plus PROC CONTENTS output:

```
data oranges;
   input variety $ flavor texture looks;
   cards;
navel 9 8 6
```

```
;
proc contents data=oranges;
run;
```

The output is shown in Output 4.3 on page 51 .

Output 4.3 CONTENTS Procedure Output

```
                               The SAS System                              1
                             CONTENTS PROCEDURE

 Data Set Name: WORK.ORANGES                        Observations:         1
 Member Type:   DATA                                Variables:            4
 Engine:        V8                                  Indexes:              0
 Created:       14:27 Monday, June 1, 1999          Observation Length:   32
 Last Modified: 14:27 Monday, June 1, 1999          Deleted Observations: 0
 Protection:                                        Compressed:           NO
 Data Set Type:                                     Sorted:               NO
 Label:

                -----Engine/Host Dependent Information-----

       Data Set Page Size:         8192
       Number of Data Set Pages:   1
       First Data Page:            1
       Max Obs per Page:           254
       Obs in First Data Page:     1
       Number of Data Set Repairs: 0
       File Name:                  ORANGES WORK A1
       Release Created             7.00.00
       Host Created                VM/ESA
       Owner Name                  USERID
            -----Alphabetic List of Variables and Attributes-----

                   #    Variable    Type    Len    Pos
                   -----------------------------------
                   2    FLAVOR      Num       8      8
                   4    LOOKS       Num       8     24
                   3    TEXTURE     Num       8     16
                   1    VARIETY     Char      8      0
```

The only values that you need to pay attention to are **`Observation Length`** and **`Compressed`**:

Observation Length
: is the record size in bytes.

Compressed
: has the value NO if records are not compressed; it has the value YES if records are compressed. (If the records are compressed, do not use the procedure given in “Estimating the Size of a SAS Data Set” on page 50 .)

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

SAS® Companion for the CMS Environment, Version 8

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

ISBN 1–58025–481–0

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, by 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 by the government is subject to restrictions as 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 DB2® 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.