New Features of the SAS® System
for Minicomputers

Stephen M. Beatrous, SAS Institute Inc.
Bruce M. Tindall, SAS Institute Inc.

I. INTRODUCTION

The SAS® system on minicomputers uses essentially the same SAS® language as the SAS® system on IBM mainframes. You can move whole applications from one type of computer to another and run them with only minor changes.

But there are differences between the two versions of the SAS® system.

Several differences are due to changes that the users asked for on the SASware Ballot. Others are due to new features that let you take advantage of facilities unique to minicomputer operating systems.

All these differences and new features for VMS SAS are covered in detail in Technical Report P-128, which is available from SAS Institute's Publications Department. (A similar report for AOS/VS SAS will be available soon.)

The topics we are going to cover briefly today are:

• How to transport SAS data sets between computers.
• How to use permanent SAS data sets and formats on minicomputers, using the LIBNAME and LIBSEARCH statements.
• Explicit indexing of SAS arrays.
• The new ATTRIB statement.
• The new SELECT statement.
• Features of external file input/output.

There is a separate session devoted to the new SAS Display Manager, so we will not cover that here.

II. TRANSPORT DATA SETS

A SAS data set may be moved from one operating system to another by first converting it to transport format. A collection of transport data sets is a transport data library. To create or read a transport data library under an IBM operating system use the COPY procedure or use the TRANSPORT= data set option.

THE XCOPY PROCEDURE -- Transport Data Sets on IBM

PROC XCOPY is used under IBM OS or VM/CMS operating systems to read and write transport data libraries. PROC XCOPY can read transport data libraries created under any of the minicomputer operating systems and can reformat SAS data sets created under IBM OS or VM/CMS so that you can use them under a minicomputer operating system.

Moving Data Sets with PROC XCOPY

SAS data sets may be transported to a minicomputer on tape or disk. The tape transport requires that the minicomputer have a tape drive and that the tape created by PROC XCOPY be of a density compatible with the minicomputer's tape drive. The disk format requires that a network exist between the IBM and the minicomputer.

Restrictions on XCOPY

XCOPY is capable of moving only one SAS data library per tape or disk file. That data library may, however, contain as many SAS data sets as are needed.

Examples of Creating a Transport Data Library with PROC XCOPY:

OS

•

EXEC SAS

/ // EXEC SAS

//

//PORT DD DSN=IXTAPE,DISP=(NEW,KEEP),
// UNIT=TAPE, VOL=SER=volumeserial, LABEL=(1,NL),
// IDC=(RECFM=FB, LRECL=80, BLKSIZE=8000, DEN=m)
// MYDATA DD DSN=A.SAS.LIBRARY,STRIP=OLD
// SYSDN *

PROC XCOPY IN=MYDATA /* LIB IN IBM FMT */

OUT=PORT /* TRANSPORT LIBRARY */

EXPORT; /* WRITE A TRANSPORT LIB */

CMS
FILEDEF MYDATA DISK SAMPLE A
FILEDEF PORT TAPI NL(RECFM FB LRECL 80 BLOCK 8000 DEN a)
OP MESSAGE ATTACH A TAPE DRIVE TO userid AS 181
OP MESSAGE MOUNT volumeserial ON THAT DRIVE
TAPE WTM (TAP1)
SAS
PROC XCOPY IN=MYDATA /* LIB IN IBM FMT */
OUT=PORT /* LIBRARY IN TRANSFER FORMAT */
EXPORT; /* WRITE A TRANSPORT */
/* DATA LIBRARY. */
Both of the above samples will convert a SAS data set denoted by the DD name MYDATA to a transportable data library on tape. That tape is referenced by the DD name PORT. The EXPORT option is used since the transport library is being written. The IMPORT option would have been used if the transport library were to be read.

Availability of XCOPY

PROC XCOPY is placed on the same tape with the SAS® product for the minicomputer sites. After installing the SAS® system the site should give the tape to their IBM counterpart who would then install XCOPY in their IBM SAS library. A tape containing only PROC XCOPY is also available for IBM sites requesting it. PROC XCOPY is not available under the minicomputer operating systems. A transport data set may be read on those systems by using PROC COPY or by using the TRANSPORT data set option.

TRANSPORT DATA SETS ON MINICOMPUTERS

A transport data set may be read or written on a minicomputer by PROC COPY or by using the TRANSPORT data set option. The libname IXTAPE is reserved for transport data libraries on magnetic tape. Any libname may be used for transport data libraries on disk. The libname for the disk will translate into a logical name similar to a DD name on an IBM operating system. A transport data set may be read on those systems by using PROC COPY or by using the TRANSPORT data set option.

Using PROC COPY to Read and Write Transport Data Sets

The keywords IMPORT and EXPORT have been added to PROC COPY to indicate that the input and output libraries respectively are in transport format.

To read a tape formatted transport data library with PROC COPY:

PROC COPY IN=IXTAPE
OUT=libname IMPORT;

To write a tape formatted transport data library:

PROC COPY IN=libname
OUT=IXTAPE IMPORT;

The advantages of using PROC COPY over the TRANSPORT data set option are that PROC COPY can write a multi-membered transport data library and is more efficient in reading a transport data library.

The limitation on PROC COPY is that only one library may be written to a tape. As many members as are required, however, may be put in that library.

Using the TRANSPORT= Data Set Option to Read and Write Transport Data Sets

The TRANSPORT= data set option specifies whether a SAS data set is in transport format.

To read a tape formatted transport data set:

DATA libname.memname;
SET IXTAPE.memname (TRANSPORT=YES);
more SAS statements or
PROC proccname
DATA=IXTAPE.memname (TRANSPORT=YES);

To write a tape formatted transport data set:

DATA libname.memname;
SET logicalname.memname (TRANSPORT=YES);
or
PROC proccname DATA=logicalname.memname;

To write a tape formatted transport data set:

DATA IXTAPE.memname (TRANSPORT=YES);
SET libname.memname;
more SAS statements or
PROC proccname
OUT=IXTAPE.memname (TRANSPORT=YES);
$ ASSIGN SASTAPE IXTAPE

The advantage of using the TRANSPORT Data Set Option over PROC COPY is that you have the flexibility of processing your data during the import or export operation. The disadvantages of using the TRANSPORT Data Set Option are that only a single member may be written to a library and that the TRANSPORT data set option...
will be slower in performing an import operation. The more members in the transport data library, the greater disparity between PROC COPY and the TRANSPORT data set option when performing an import operation.

Example of Reading a Transport Data Library Under VMS

On each operating system there will be some host level commands necessary to perform I/O on a transport data set. The tape must be mounted and dismounted if the data set resides on tape. A link between the logical name used as the libname and the physical data set must be made. The following example will read a tape formatted transport data library. The transport library contains the members A, B, and C.

```
$ ALLOCATE IXTAPE
$ REQUEST "PLEASE MOUNT VOLUME TBNDAT01"
$ MOUNT/FOREIGN IXTAPE
$ SAS
?
LIBNAME MYDATA [BEATROUS.IBMSTUFF];
*/ THE MEMBERS A, B, AND C WILL */
*/ BE TRANSLATED INTO VMS FORMAT*/
? PROC COPY IN=IXTAPE
? OUT=MYDATA
? IMPORT;
*/ ------IMPORT------ */
*/ TRANSPORT DATA SET OPTION */
? DATA MYDATA.ENTRY;
*/ SET IXTAPE.A(TRANSPORT=YES);
*/ IF GROUP=1;
?
$ DISMOUNT IXTAPE
$ REALLOCATE IXTAPE
```

III. LIBNAME AND LIBSEARCH

PERMANENT SAS DATA SETS AND FORMATS

Once you have moved your existing SAS data sets to your minicomputer, you will probably be creating and using permanent SAS data sets and permanent formats.

TWO-LEVEL DATA SET NAMES

As you already know, permanent SAS data sets are usually referred to by a two-level name, like this:

```
DATA ABC.DEF
```

The second level ("DEF" in our example) is the name of the SAS data set.

On minicomputers, the first level ("ABC" in our example) stands for the directory in which the SAS data set is located. There is a new SAS statement that associates a SAS first-level data set name with a directory name.

THE LIBNAME STATEMENT

```
LIBNAME ABC [HOLLY.WOOD];
```

is an example of the new LIBNAME statement. It associates the SAS name ABC with a VMS directory named [HOLLY.WOOD].

After that LIBNAME statement is executed, the statements

```
DATA ABC.DEF;
SET ABC.XYZ;
```

will cause the SAS® system to create a SAS data set named DEF, in the directory [HOLLY.WOOD], and to read from a SAS data set named XYZ, in the same directory.

FORMATS AND THE LIBNAME STATEMENT

The LIBNAME statement also tells the SAS® system where to store permanent formats that you create with PROC FORMAT.

For example, these statements create a format called YESNO and store it in the directory [HOLLY.WOOD]:

```
LIBNAME MYFMTS [HOLLY.WOOD];
PROC FORMAT DDNAME=MYFMTS;
VALUE YESNO 1 = 'YES'
2 = 'NO';
```

LIBSEARCH

With the LIBSEARCH statement you can give the SAS® system a list of one or more directories in which to look for input SAS data sets and for permanent user-defined formats. (AOS/VS users can see the analogy to the AOS/VS SEARCHLIST command.)

After you have defined one or more LIBNAMEs, you can use them in a LIBSEARCH statement.

For example:

```
LIBNAME A ['CARS.1984'];
B ['CARS.1983'];
C ['CARS.1982'];
LIBSEARCH A B C;
DATA;
SET FORD;
```
In the example above, we use a ONE-LEVEL name for an input SAS data set ("FORD"). Because we have specified a LIBSEARCH list, the SAS® system will search those three directories for a SAS data set named FORD, and will use the one it finds first.

If none of the directories in the LIBSEARCH list contains the data set named FORD, then the SAS® system will try to use the temporary SAS data set WORK.FORD.

If you specify a particular directory, like this:

```sas
DATA;
SET B.FORD;
```

then the SAS® system will ignore the LIBSEARCH list.

To CREATE a permanent SAS data set, you must still specify a two-level name in the DATA statement. LIBSEARCH affects only input SAS data sets, not output SAS data sets.

FORMATS AND THE LIBSEARCH STATEMENT

The LIBSEARCH statement also tells the SAS® system where to find permanent, user-defined formats.

A moment ago we showed how to create a permanent format called YESNO in the directory [HOLLY.WOOD]. If you want to use that format in a later SAS job, use these statements:

```sas
LIBNAME F '[HOLLY.WOOD]';
LIBSEARCH F;
DATA;
  INPUT X;
  PUT X YESNO.;
```

The LIBSEARCH command causes the DATA step to search the directory represented by the LIBNAME "F" for the format "YESNO". You can, of course, concatenate several directories for this purpose, by specifying more than one LIBNAME in the LIBSEARCH statement.

In this respect, LIBSEARCH is similar to the SASLIB option of CMS SAS.

IV. ARRAYS

EXPLICIT INDEXING OF ARRAYS

In the 1983 SASware Ballot, the most-requested new feature was "explicit indexing of arrays." This feature is now available in the SAS® system on minicomputers.

DEFINING AN ARRAY

Use the ARRAY statement to define an array. There has been no change to the ARRAY statement, except that the index variable is never needed.

To define an array named ARR that contains the variables A, B, C1, C2, C3, C4, and C5, use the statement:

```sas
ARRAY ARR A B C1-C5
```

USING AN ARRAY

When you use an element of that array later in the DATA step, specify the index explicitly, in brackets or braces after the array name. For example:

```sas
I = ARR[2]; /* Second element of ARR */
ARR[Y] = J; /* Y'th element of ARR */
I = ARR[A+B]; /* (A+B)'th element of ARR */
```

PROCESSING ALL ELEMENTS OF AN ARRAY

A new DATA step function, DIM (for "dimension"), helps you process all elements of an array.

If ARR is an array, DIM(ARR) is the number of elements in the array. The following SAS statements process every element of the array ARR, changing missing values to zeros:

```sas
DO I = 1 TO DIM(ARR);
  IF ARR[I]=. THEN ARR[I]=0;
END;
```

TWO-DIMENSIONAL ARRAYS

Two-dimensional arrays will be available in later releases of the SAS® system. To define a two-dimensional array with 10 rows and 5 columns, use the statement:

```sas
ARRAY X(10,5) X1-X50
```

The 23rd element of this array - the variable X23 - is X[2,3].

V. ATTRIB

The ATTRIB statement allows you to specify all of a variable's attributes in a single data step statement.
With the ATTRIB statement can specify one or more of the following attributes:

- format
- informat
- length
- label

For example, the statement

```
ATTRIB BDAY FORMAT=DATE7. LABEL='DATE OF BIRTH' NAME LENGTH=$10;
```

assigns a format and a label to the variable BDAY and types the variable NAME as character with a length of 10 bytes.

## VI. SELECT STATEMENT

The SELECT statement allows one of several blocks of code to be executed depending on the value of a control expression. WHEN clauses define the conditional blocks of code for the SELECT statement. An optional OTHERWISE clause defines a block of code to be executed if none of the WHEN conditions are satisfied.

An example of a SELECT group appears below:

```
DATA CONTROL GROUP1 GROUP2 GROUP3;
/*-----------------------------------*/
/* THIS DATA STEP WILL SPLIT THE SAS */
/* DATA SET "EXPDATA" INTO FOUR */
/* COMPONENT DATA SETS USING A */
/* SELECT GROUP. */
/*-----------------------------------*/
SET EXPDATA;
SELECT ( GROUP );
WHEN ( 'CONTROL' ) OUTPUT CONTROL;
WHEN ( 'GROUP1' ) OUTPUT GROUP1;
WHEN ( 'GROUP2' ) OUTPUT GROUP2;
WHEN ( 'GROUP3' ) OUTPUT GROUP3;
OTHERWISE DO;
PUT "UNKNOWN GROUP:' GROUP;
ABORT ABEND;
END;
/* SELECT GROUP */
```

## VII. EXTERNAL IO

External files (files that are not SAS data sets) can be read and written easily using the SAS® system. Basically, a logical name assignment is performed and that logical name is used on a INFILE or FILE statement. The technique is similar to using a DDNAME under an IBM operating system.

### Accessing External Files under VMS

```
X 'ASSIGN OUTCH.DAT SUNBURN';
DATA _NULL_;
FILE SUNBURN;
BURN = 0; COLOR = 'BROWN';
PUT COLOR BURN;
BURN = 1; COLOR = 'PINK';
PUT COLOR BURN;
BURN = 2; COLOR = 'RED';
PUT COLOR BURN;
RUN;
DATA _NULL_;
INFILE SUNBURN;
INPUT COLOR $ BURN;
RUN;
```

### Format Alignment Specifications

Formats in the SAS® system for minicomputers may appear with alignment or justification specifications. This feature allows you to left justify, center, or right justify a variable in an output field.

The following program uses the -C format alignment specification to center output in a 70 byte field and uses the -L specification to left justify a output in a 15 byte field.

```
DATA _NULL_;
LINE1 = '**************************';
LINE2 = 'MIDDLE OF THE ROAD';
PUT /* LINE1 $70.-C / */
LINE2 $70.-C /
LINE1 $70.-C;
SALARY = 1000;
PUT 'SALARY=' SALARY DOLLAR15.2-L;
```

The above program produces the following output:

```
************************** MI DDLE OF THE ROAD
SALARY ~ $1,000.00
```

### Reading Data Sensitive Files

The most common file structure on minicomputers is data sensitive. Files created with a text editor, for example, would be data sensitive. Data sensitive files contain varying length records which are terminated by a special record termination character. These files present a unique problem in applications where it is necessary to read an entire record into a SAS variable.

Example: Write a DATA STEP to read varying length records from a data sensitive file into a single variable.
Known:
- The format of the file (data sensitive).
- The maximum length of the records in the file (80).

DATA NULL;
/* L will contain each line length. */
INFILE REPORT LENGTH=L;
/* Dummy input to fetch length. */
INPUT L;
/* Reread the current line. */
/* Read "T: " char into "record". */
INPUT RECORD VARYING80. L;