SAS User File for H16IF1 Data

This file contains information and sample SAS programs to create a permanent SAS dataset, for users who want to use SAS in processing the MEPS data provided in this PUF release.

There are two ways to create a permanent SAS dataset, using either the SAS transport data file (H16IF1.SSP) or the ASCII data file (H16IF1.DAT) supplied in this PUF release. Section A provides a sample SAS program for the first alternative, which is to convert the SAS transport data file to a regular SAS dataset, using the SAS PROCedure: XCOPY. Section B provides a sample SAS program for the second alternative, which is to read data from the ASCII data file, using a SAS DATA step with INFILE, INPUT, and LABEL statements. Section C explains format-related SAS statements that a user may optionally use when working with the SAS dataset. Examples of SAS programs (DATA step or PROC) are provided in all three sections, primarily for the benefit of inexperienced users. Section D contains complete SAS statements that must be used in the programs described in Sections B and C.

****

The sample SAS programs provided in Sections A and B show how to create a permanent SAS dataset from the data files provided in this PUF release.

A. A Sample SAS Program for Converting the SAS Transport File to a Permanent SAS Dataset

The SAS transport file H16IF1.SSP was created from a PC-SAS Version 6.12 data file, using PROC COPY. This file has been tested for use with PC-SAS version 6.10 or higher, or with mainframe SAS version 6.08 or higher. This file may work with earlier versions of SAS, although it has not been tested with those versions. Users who are unable to use this SAS Transport file should instead convert the ASCII data file H16IF1.DAT to a SAS data set as described in Section B.

The SAS PROCedure XCOPY will read a SAS transport data file and convert the data to regular SAS format, storing the output in a permanent SAS dataset. This permanent SAS dataset can then be used for all future processing and analyses.

Below is a sample PC-SAS program that can be used to convert the SAS transport file to a permanent PC SAS dataset (in a Windows environment, with SAS V6.12).

```
     LIBNAME PUFLIB 'C:\MEPS';
     FILENAME IN1 'C:\MEPS\H16IF1.SSP';

     PROC XCOPY IN=IN1 OUT=PUFLIB IMPORT;
     RUN;
```

If the user wants a list of variables and a few sample records in the permanent SAS dataset, following are the SAS statements to accomplish these.

```
     PROC CONTENTS DATA=PUFLIB.H16IF1;
     TITLE "List of Variables in MEPS H16IF1 SAS Dataset";
     RUN;

     PROC PRINT DATA=PUFLIB.H16IF1 (OBS=20);
     TITLE "First 20 Observations in MEPS H16IF1 SAS Dataset";
     RUN;
```

The LIBNAME statement tells SAS the location (directory name) to store the permanent SAS dataset which is output by PROC XCOPY. The FILENAME statement tells SAS the location (complete directory and file name) of the input SAS transport data file.

NOTES:

1) The names used in the LIBNAME and FILENAME statements shown above (i.e., PUFLIB, IN1) are arbitrary; they are only temporary aliases.

2) The directory and file names used in the LIBNAME and FILENAME statements shown above are Windows syntax and may need to be modified for other operating systems such as UNIX, MAC/OS, VMS, or OS/2.

3) H16IF1 is the internal SAS dataset name (also the PC file name, without the extension) prior to the creation of the SAS transport data file. After running PROC XCOPY, the output SAS dataset assumes the same dataset name (or file name). Hence, in the example above, a file named H16IF1.SD2 will be created under the C:\MEPS directory when PROC XCOPY runs successfully.

B. A Sample SAS Program for Converting the ASCII Data File to a Permanent SAS Dataset

The complete SAS statements (INPUT and LABEL) included in Section D are intended to save time for those users wishing to create a permanent SAS dataset from the H16IF1.DAT ASCII data file. These statements must be used in combination with other SAS statements to create the appropriate SAS program, as shown below. To use the statements provided in Section D to create a SAS program, you will need an ASCII text editor. If you are using an interactive form of SAS (Windows, UNIX, OS2, etc.), use the editor provided as part of the SAS software.

Following is a sample SAS program that will convert the ASCII data file to SAS format.

```
     LIBNAME PUFLIB 'C:\MEPS';
     FILENAME IN1 'C:\MEPS\H16IF1.DAT';

     DATA PUFLIB.H16IF1;
     INFILE IN1 LRECL=57;
     INPUT .....; * to user: insert the complete INPUT statement that is
provided in Section D;
     LABEL .....; * to user: insert the complete LABEL statement that is
provided in Section D;
     RUN;
```

Here is an explanation of the SAS statements used in the program above.

LIBNAME statement: This tells SAS the location (directory name) of the permanent SAS dataset.

FILENAME statement: This tells SAS the location of the input ASCII data file.

DATA statement: This signifies the beginning of a SAS DATA step and specifies describes the output SAS dataset, referencing the LIBNAME entry (PUFLIB) and assigning an internal SAS dataset name (H16IF1). In the example, after the successful completion of the DATA step, a PC file named H16IF1.SD2 would have been created in the C:\MEPS directory.

INFILE statement: This tells SAS the location (directory and file name) of the input ASCII data file. Also provided is the logical record length (57 bytes), with the default of RECFM=V implied when this parameter is omitted. LRECL and RECFM are optional parameters in the INFILE statement. With regard to these options, please note the following: the ASCII data file H16IF1.DAT contains a 2-byte carriage return/line feed at the end of each record. When converting to a PC-SAS file, the LRECL option should be used to specify the record length to avoid use of a default record length by PC-SAS. If the RECFM=V option is used, the LRECL option must be specified as the logical record length (e.g., 57 for H16IF1.DAT). If RECFM=F is used, then the LRECL value must be specified as the logical record length plus 2 (59 for H16IF1.DAT). Note that if the RECFM option is omitted, then the default option of RECFM=V is automatically used, and LRECL should be specified as the logical record (57 for H16IF1.DAT).

INPUT statement: This specifies the input record layout, giving names and the beginning and ending column positions for data items (which become SAS variables) in the ASCII data file (H16IF1.DAT). Variable type (numeric or character) is also defined via the INPUT statement.

LABEL statement: This associates descriptive names with the SAS variables.

RUN statement: This tells SAS to execute all commands up to this point.

C. Optional Format-related SAS Statements

If a user wants to use formats for the SAS variables, a SAS format library must first be created. Below is a SAS program that will accomplish this.

```
     LIBNAME PUFLIB 'C:\MEPS';

     PROC FORMAT LIBRARY=PUFLIB;
     VALUE .....; * to user: insert the complete set of VALUE statements found
in Section D;
     VALUE .....;
     ......... ;
     RUN;
```

Below is an example of how to use the SAS formats defined by the PROC FORMAT procedure.

```
     LIBNAME PUFLIB 'C:\MEPS';
     OPTIONS FMTSEARCH=(PUFLIB);

     PROC FREQ DATA=PUFLIB.H16IF1;
     TABLES .... / LIST MISSING;
     FORMAT varnaml fmtnaml.  Varnam2 fmtnam2.  .... ;
     * to user: substitute varnaml and fmtnaml with actual variable names and
format names;
     *     Insert the FORMAT statement provided in Section D, if you are using
all the variables;
     *     in the TABLES statement;
     TITLE "Frequency Distributions ....";
     RUN;
```

Here is an explanation of the SAS statements used above.

LIBNAME statement: This tells SAS the location (directory name) of the SAS format library. Please note that SAS datasets (file name extension is 'SD2') and format libraries (file name extension is 'SC2') can be stored under the same directory.

OPTIONS FMTSEARCH=...: This specifies the SAS format library.

PROC FORMAT statement: This identifies the SAS procedure that will make SAS formats according to VALUE statements. Formats will be stored in a file named FORMATS.SC2. Please note that the option 'LIBRARY=...' can be omitted, if the user does not want to create a permanent SAS format library. When simply 'PROC FORMAT;' is used, the formats are defined only for the duration of the batch SAS program or an interactive SAS session.

VALUE statement: This gives a) names to formats; and b) descriptive labels for individual values, or range of values. The format names can then be invoked using a FORMAT statement, if desired.

PROC FREQ statement: This identifies the SAS procedure that will generate frequency distributions of variables specified in the TABLES statement, formatted if a FORMAT statement is used. The input SAS dataset is specified in the 'DATA=' option.

FORMAT statement: This associates existing formats with variables. When using this statement, the formats must have already been created with a PROC FORMAT procedure.

RUN statement: This tells SAS to execute all commands up to this point.

NOTES:

1) Use of formats is entirely optional, and depends on the type of analyses that you are doing. It is recommended that you create and use them as appropriate.

2) The names used in the LIBNAME and FILENAME statements shown above (i.e., PUFLIB, IN1) are arbitrary; they are only temporary aliases.

3) You only create the permanent SAS data set once. Additional analyses can be run using this permanent dataset.

4) The file and directory specifications in the LIBNAME and FILENAME statements are Windows syntax and may need to be modified for other operating systems such as UNIX, MAC/OS, VMS, or OS/2.

D. SAS Statements

This section contains SAS INPUT, LABEL, FORMAT and VALUE statements for use in converting the ASCII H16IF1.DAT file into a SAS data set, and for creating SAS formats.

```
* INPUT STATEMENTS;
INFILE IN LRECL=57;

INPUT @1      DUPERSID $8.0
      @9      CONDIDX  $12.0
      @21     EVNTIDX  $12.0
      @33     CLNKIDX  $24.0
      @57     EVENTYPE  1.0
;


* FORMAT STATEMENTS;
FORMAT DUPERSID $VALIDID.
       CONDIDX  $VALIDID.
       EVNTIDX  $VALIDID.
       CLNKIDX  $VALIDID.
       EVENTYPE EVENTYPE.
;


* LABEL STATEMENTS;
LABEL DUPERSID='PERSON ID: DUID+PID'
      CONDIDX ='CONDITION ID'
      EVNTIDX ='EVENT ID'
      CLNKIDX ='CLNK ID: CONDIDX+EVNTIDX'
      EVENTYPE='TYPE OF EVENT CONDITION IS LINKED TO'
;


* VALUE STATEMENTS;
VALUE EVENTYPE
  1 = '1 MVIS'
  2 = '2 OPAT'
  3 = '3 EROM'
  4 = '4 STAZ'
  5 = '5 DVIS'
  7 = '7 HVIS'
  8 = '8 PMED'
  ;
VALUE $VALIDID
  '0' < - HIGH = 'VALID ID'
  ;
```