

SAS® to .dbf by Way of MVS®

Robert F. Davis
Micro Craft Technology/AEDC Operations
Arnold Engineering Development Center
Arnold Air Force Base, TN 37389

ABSTRACT

Database applications such as dBaseIV® and FOXPRO® require that a data structure be defined before the actual data can be input. Downloading data from the mainframe computer (IBM® 3090) in a flat file does not provide this data structure. The Macro "SASTODBF" described in this paper creates the data structure and the data in one file, ready for use by these applications. The Macro SASTODBF takes as its input an SAS data set and produces a binary output file corresponding to a PC .dbf file. The SAS data set can be created from existing SAS data or from external flat files such as ADR® DATAQUERY® export files. This solves the problem of getting data from the mainframe computer into a PC database application in a timely manner.

INTRODUCTION

This Macro was designed on an IBM 3090 mainframe running the MVS operating system. The Macro was developed to assist PC users at Arnold Engineering Development Center (AEDC) in getting data from the mainframe computer in a convenient form to use in their PC-based database applications. The output file produced by this Macro results in a direct binary image of a .dbf file. This file may be downloaded from the 3090 mainframe to a PC using a binary file transfer, and may be used directly by any database application that uses .dbf file format. This Macro considers only .dbf data types for character, numeric, and date. Other .dbf data types such as memo and logical could be programmed but are not discussed here because SAS does not normally use these data types. All SAS variable values must fall within the minimum and maximum values for the .dbf data type with which they are associated. Figure 1 is an example use of the Macro. Figure 2 is the Macro definition.

SASTODBF MACRO (analysis)

This Macro takes as input two arguments. The first argument is the name of the SAS data set to be converted. The second is the name of the binary output file to be created. One

restriction applies to the SAS input data set. SAS date values are considered to be numeric data types and cannot be distinguished from other numbers. In order to identify dates for the conversion to .dbf, all date variables must have a format of YYMMDD. assigned to them. If a date has been defined in the SAS data set as a character variable, it must first be converted to an SAS date value (numeric variable) with a YYMMDD. format. This date formatting can be done by a preprocess data step before calling the SASTODBF MACRO. Another possible preprocess step could be to use attribute or retain statements to arrange the variables in a specific order. The first part of this macro code defines another macro called NUMOBS. NUMOBS' purpose is to get the number of observations in an SAS data set specified by argument DSN and store it in a global variable name specified by the argument VAR. Next, PROC CONTENTS is executed on the input data set to get the names and format information of all the variables. These contents are then sorted by the variable number that was assigned to each variable when the data set was first created. The macro NUMOBS is called for the contents data set to get the total number of variables in the original data set. This is stored at global variable NO_VARS. A null data step is now processed using the contents data set. Using the data type, format, format length, and length variables, this data step calculates the total width of all variables and stores it in global variable TOT_LEN. The NUMOBS macro is called again, this time using the original input data set to get the total number of observations. This number is stored in global variable TOTALOBS. The next data step is in two parts. The first part executes only on the first time through the data step to create the base header of the .dbf file. The current date along with the previously obtained header length and record length are formatted and output with a single put statement. A trailing @ symbol is used to hold the output because the entire header record must be written as one record. The second part of this data step continues the output by creating the field descriptors or field subrecords. A 32-byte block for each variable containing the variable name, data type, format lengths, and null fillers for each variable from the input data set is continuously output followed by a hexadecimal '0d' after the last variable to terminate the header record. The last

data step in this macro definition is a source code generator. Using the data from the input data set, this data step generates SAS code to a work file. This code, when executed, appends to the .dbf header record data all the variable values in their proper formats and completes writing the .dbf file by outputting the .dbf data terminator a hexadecimal '1A'. After this code is written to a work file, the file is then rewound. This rewind takes the form of a FILENAME clear statement and reissuing the FILENAME statement defining the file again. Finally a %INCLUDE statement is issued to execute the generated code.

CONCLUSION

This MACRO has been executed on an IBM 3090 mainframe computer running SAS version 6.08 under MVS operating system. The resultant output has been tested using dBaseIV, and FOXPRO database applications.

REFERENCES

SAS Language Reference, Version 6 First Edition.
SAS Procedures Guide, Version 6 Third Edition.
SAS Guide to Macro Processing, Version 6 Second Edition

ACKNOWLEDGMENTS

SAS is a registered trademark of SAS Institute, Inc., SAS Campus Drive, Cary, NC 27513

dBase IV is a registered trademark of Borland International, Inc.

Foxpro is registered trademark of Microsoft Corporation in the United States and other countries.

```

/****EXAMPLE USE OF SASTODBF *****/
FILENAME DBF 'T1.SESUG95.DBF' RECFM=VB
DSORG=PS LRECL=4100 BLKSIZE=20504;
DATA SAMPLE;
ATTRIB JOB_NO LENGTH=4.;
ATTRIB ENT_DATE FORMAT=YYMMDD10.
INFORMAT=MMDDYY6.;
ATTRIB TITLE LENGTH=$35;
ATTRIB FY LENGTH=4.;
ATTRIB PHASE LENGTH=$2;
ATTRIB COST FORMAT=DOLLAR10.;
ATTRIB MAN_HRS LENGTH=6.;

```

```

INFILE CARDS DLM='';
INPUT JOB_NO ENT_DATE TITLE
FY PHASE COST MAN_HRS;
CARDS;
0001,022095,"PROJECT A",1994,P,230000,100
0001,022895,"PROJECT A",1995,D,150000,200
0001,030195,"PROJECT A",1996,NP,2500000,12000
0002,031095,"PROJECT B",1997,P,80000,250
;
%SASTODBF(SAMPLE,DBF);
QUIT;

```

FIGURE 1. SASTODBF EXAMPLE USAGE.

Host Systems

```

%MACRO SASTODBF(SASDATA,OUTFILE);
/*****/
/* SOURCE CODE NAME: SASTODBF */
/* COMPANY: MICRO CRAFT TECHNOLOGY */
/* AUTHOR: ROBERT F. DAVIS */
/* PURPOSE: */
/* THIS MACRO DOES A DIRECT CONVERSION */
/* OF A SAS DATA SET TO AN MVS DATA SET */
/* IN PC .DBF FORMAT. */
/* REQUIRES: */
/* INPUT DATA SET SPECIFIED BY ARGUMENT */
/* SASDATA. DATE VARIABLES MUST HAVE A */
/* FORMAT OF 'YYMMDD' TO IDENTIFY THEM */
/* AS .DBF DATE VALUES. */
/* OUTPUT: */
/* OUTPUTS A BINARY FILE OF A SAS DATA */
/* SET IN A PC .DBF FORMAT TO THE */
/* ARGUMENT SPECIFIED BY OUTFILE. */
/* */
/*****/
/* MACRO TO GET THE TOTAL NUMBER OF
RECORDS IN A SAS DATA SET
ARGUMENT 1=INPUT SAS DATA SET NAME (DSN)
ARGUMENT 2=GLOBAL COMMON VARIABLE NAME IN
WHICH TO STORE THE TOTAL NUMBER
OF RECORDS (VAR) */
%MACRO NUMOBS(DSN,VAR);
%GLOBAL &VAR;
DATA _NULL_;
RETAIN NUMBER 1;
IF 0 THEN SET &DSN POINT=NUMBER NOBS=COUNT;
CALL SYMPUT("&VAR",LEFT(PUT(COUNT,8.)));
STOP;
RUN;
%MEND NUMOBS;
/* GET THE CONTENTS OF THE SAS DATA SET */
PROC CONTENTS DATA=&SASDATA OUT=CONTENTS NOPRINT;
PROC SORT DATA=CONTENTS;
BY VARNUM;
/* GET THE TOTAL NUMBER OF VARIABLES
IN THE SAS DATA SET */
%NUMOBS(CONTENTS,NO_VARS)
/* CALCULATE THE TOTAL LENGTH OF ALL THE
VARIABLES AND STORE IT IN GLOBAL COMMON */
DATA _NULL_;
RETAIN TOT_LEN 0;
SET CONTENTS END=EOF;
/* FORMATTED NUMERIC VARIABLE */
IF (TYPE EQ 1 AND FORMATL NE 0 AND
FORMAT NE 'YYMMDD') THEN TOT_LEN+FORMATL;
/* CHARACTER OR DATE VARIABLE OR
NON-FORMATTED NUMERIC VARIABLE */
ELSE TOT_LEN+LENGTH;
IF (EOF) THEN
CALL SYMPUT('TOT_LEN',TOT_LEN);
RUN;
/* GET TOTAL NUMBER OF RECORDS

```

```
IN THE SAS DATA SET */
%NUMOBS(&SASDATA,TOTALOBS);
RUN;
DATA OUT1;
/* .DBF BASE HEADER DESCRIPTION */
ATTRIB VER LENGTH=2 LABEL='VERSION';
ATTRIB YY LENGTH=2 LABEL='YEAR';
ATTRIB MM LENGTH=2 LABEL='MONTH';
ATTRIB DD LENGTH=2 LABEL='DAY';
ATTRIB NUM_RECS LENGTH=4 LABEL='NUMBER OF RECORDS';
ATTRIB HEAD_LEN LENGTH=2 LABEL='HEADER LENGTH';
ATTRIB REC_WIDT LENGTH=2 LABEL='RECORDWIDTH';
ATTRIB FILLER LENGTH=$20 LABEL=NULL FILLER';
ATTRIB NAME_OUT LENGTH=$11 ;
RETAIN FILLER
'00000000000000000000000000000000000000000000X';
RETAIN VER 03X;
RETAIN BASE 32 SUB_SIZE 32;
SET CONTENTS END=EOF;
FILE &OUTFILE;
IF (_N_=1) THEN DO;
/* CREATE THE BASE HEADER INFORMATION AND OUTPUT IT */
DATE=TODAY(); /* CURRENT DATE */
YY=MOD(YEAR(DATE),100);/*CURRENT YEAR*/
MM=MONTH(DATE); /* CURRENT MONTH */
DD=DAY(DATE); /* CURRENT DAY */
/*TOTAL NUMBER OF RECORDS IN THE DATA SET TO BE CONVERTED */
NUM_RECS=&TOTALOBS;
/* CONVERT 32 BIT INTEGER TO 4 BYTES */
NO_REC4=INT(NUM_RECS/16777216);
NUM_RECS=MOD(NUM_RECS,16777216);
NO_REC3=INT(NUM_RECS/65536);
NUM_RECS=MOD(NUM_RECS,65536);
NO_REC2=INT(NUM_RECS/256);
NO_REC1=MOD(NUM_RECS,256);
/* LENGTH OF THE HEADER TO BE CREATED */
HEAD_LEN=BASE+((SUB_SIZE*&NO_VARS)+1);
/* CONVERT 16 BIT INTEGER TO 2 BYTES */
HD_LENHI=INT(HEAD_LEN/256);
HD_LENLO=MOD(HEAD_LEN,256);
/* .DBF DATA RECORD LENGTH */
/* PLUS ONE FOR DELETE BYTE */
REC_WIDT=&TOT_LEN+1;
/* CONVERT 16 BIT INTEGER TO 2 BYTES */
REC_WDHI=INT(REC_WIDT/256);
REC_WDLO=MOD(REC_WIDT,256);
/* OUTPUT THE BASE HEADER BLOCK */
PUT VER PIB1.
YY PIB1. MM PIB1. DD PIB1.
NO_REC1 PIB1. NO_REC2 PIB1.
NO_REC3 PIB1. NO_REC4 PIB1.
HD_LENLO PIB1. HD_LENHI PIB1.
REC_WDLO PIB1. REC_WDHI PIB1.
FILLER $ASCII20. @;
END;
```

FIGURE 2. SASTODBF MACRO DEFINITION.

```

/* CREATE THE FIELD SUB RECORD DATA AND OUTPUT IT */
/* VARIABLE NAME */
NAME_OUT= TRIM(NAME) || FILLER;
/* DATE VARIABLE TYPE */
IF (FORMAT EQ 'YYMMDD') THEN DB_TYPE='D';
/* NUMERIC VARIABLE TYPE */
ELSE IF (TYPE EQ 1) THEN DB_TYPE='N';
/* CHARACTER VARIABLE TYPE */
ELSE DB_TYPE='C';
/* DEFAULT DBASE DATE LENGTH */
IF (FORMAT EQ 'YYMMDD') THEN DO;
  WIDTH=8;
  DECIMALS=0;
END;
/* NUMERIC DATA FORMAT LENGTH */
ELSE IF (TYPE EQ 1 AND FORMATL NE 0) THEN DO;
  WIDTH=FORMATL;
  DECIMALS=FORMATD;
END;
/* CHARACTER DATA LENGTH */
/* OR NUMERIC DATA WITH NO FORMAT */
ELSE DO;
  WIDTH=LENGTH;
  DECIMALS=0;
END;
/* OUTPUT FIELD SUBRECORD DATA (VARIABLE NAME,
TYPE AND FORMAT LENGTH) */
PUT NAME_OUT $ASCII1. DB_TYPE $ASCII1.
  FILLER $ASCII4.
  WIDTH PIB1. DECIMALS PIB1.
  FILLER $ASCII14. @;
/* OUTPUT HEADER TERMINATOR */
IF (EOF) THEN PUT '0D'X;
FILENAME STEP 'T1.SESUG95.DAT';
/* CREATE PROGRAM SOURCE CODE TO OUTPUT
THE CONTENTS OF THE DATA SET */
DATA SOURCE;
  RETAIN NUMOBS;
  RETAIN B -1;
  SET CONTENTS END=EOF;
  FILE STEP;
  IF (N_=1) THEN DO;
    PUT 'DATA OUT2;';
    PUT 'ATTRIB VALUE LENGTH=$20;';
    PUT 'SET &SASDATA END=EOF;';
    PUT 'FILE &OUTFILE MOD;';
    PUT 'PUT '20'X @;';
    NUMOBS=NOBS;
  END;
  /* CREATE SAS STATEMENTS TO
  OUTPUT NUMERIC DATA */
  IF (TYPE EQ 1) THEN DO;
    IF (FORMAT EQ 'YYMMDD') THEN DO;
      PUT 'IF (" NAME "EQ .) THEN VALUE=',';';
      PUT 'ELSE VALUE=LEFT(PUT(YEAR("NAME +B "),Z4.)) || '
        'LEFT(PUT(MONTH(" NAME +B "),Z2.)) || '
        'LEFT(PUT(DAY(" NAME +B "),Z2.));';
      /* DATE VALUE. */
      PUT 'PUT VALUE $ASCII8. @;';
    END;
    ELSE IF (FORMATL NE 0) THEN DO;
      PUT 'IF (" NAME "EQ .) THEN VALUE=',';';
      PUT 'ELSE VALUE=PUT("NAME +B ", " FORMATL +B
        ' " " FORMATD +B ");';';
      /* NUMERIC VALUE WITH FORMAT */
      PUT 'PUT VALUE $ASCII " FORMATL +B " . @;';
    END;
    ELSE DO;
      PUT 'IF (" NAME "EQ .) THEN VALUE=',';';
      PUT 'ELSE VALUE=PUT(" NAME +B ", " LENGTH +B ");';';
      /* NUMERIC VALUE WITH NO FORMAT */
      PUT 'PUT VALUE $ASCII " LENGTH +B " . @;';
    END;
  END;
  /* CHARACTER DATA */
  PUT 'PUT " NAME " $ASCII " LENGTH +B " . @;';
  /* OUTPUT .DBF DATA TERMINATOR */
  IF (EOF) THEN DO;
    PUT 'IF (EOF) THEN PUT '1A'X;';
  END;
RUN;
FILENAME STEP CLEAR;
/* REWIND THE PROGRAM FILE */
DATA DUMMY;
  FILENAME STEP 'T1.SESUG95.DAT';
  RUN;
  /* RUN THE CREATED PROGRAM AND OUTPUT THE
  DATA SET DATA APPENDING IT TO THE
  HEADER DATA. */
  %INCLUDE STEP;
  RUN;
  %MEND SASTODBF;

```

FIGURE 2. CONCLUDED.