

PROVIDING USER FRIENDLY ACCESS AND SUBSETTING CAPABILITIES TO NON-SAS DATA ACROSS OPERATING PLATFORMS.

Jim Anderson, Blue Cross Blue Shield of Nebraska

Advanced Macro techniques, SAS/CONNECT® software and a SAS® software Data Step View are used to span operating platforms, access remote files and create locally resident SAS Data Sets. Macro Windows provide for the specification of remote external data files for input, the entry and editing of values used to subset the remote files and the naming of local output data sets. Automated connection facilitates the transfer of selection criteria from the local to the remote platform. The remote environment utilizes the received criteria to direct SAS Macro access to and selection from external data files using a SAS Data Step View. SAS data are downloaded to the local processor without the creation of intermediate remote data sets. Following automated SAS/CONNECT signoff, control returns locally.

FRIENDLY

With the proliferation of computing technology in the workplace, there exists a large population of non-programmers who must advance beyond the reception of canned reports to interactive data analysis. A growing number of SAS products are addressing the needs of the non-programmer. These sophisticated SAS products incur additional licensing fees and are therefore not available for all sites. However, Base SAS software is required for all SAS installations and, using the macro facility, a developer may create "windowed" systems that are impressively friendly to the user and flexible for the application.

THE SITUATION

Local platform SAS data sets are to be created by subsetting data that are resident on a remote platform and are in the native format of the remote environment. The system must accommodate use by an employee with extensive and specialized business knowledge but no background in SAS or computer programming. Base SAS software and SAS/CONNECT are installed on both platforms.

REMOTE DATA

It will be necessary to read source data that are external to the SAS software system and are in the native format. The data exists as files that are individual by transaction month. A naming convention specifies that the last four characters of a file name will identify the year and month (YYMM). This convention is significant in that it allows for file access using the iterative %DO statement. Individual files of monthly data are accessible through the resolution of the %DO statement's index variable.

```
%DO I=9301 %TO 9306;  
  DATA SET&I;  
    INFILE FILE&I;  
    ...  
  RUN; %END;
```

ALLOCATION

Data files and SAS libraries must be allocated in some fashion prior to their use. Dependent upon the level of computing expertise of the intended user, it may be of benefit to make allocation as transparent as possible. Placements of allocation statements in a autoexec file or assignment to function keys are alternatives. When using the SAS macro facility, the Macro Autocall and Stored Compiled Macro Facilities are added options.

AUTOEXEC

Upon invocation, the SAS System searches for an Autoexec file and automatically executes its contents of valid SAS statements. An Autoexec file is specified as a system option and the syntax for file specification are system dependent.

A FILENAME statement placed inside an autoexec file will execute upon SAS invocation. There is no need for the non-programmer to have knowledge of the FILENAME syntax.

```
DOS:  
FILENAME PROGRAM 'C:\MYSAS\PROGRAM.PGM';
```

```
WINDOWS:  
FILENAME DIRECT 'C:\MYSAS';  
FILENAME PROGRAM 'C:\MYSAS\PROGRAM.PGM';
```

```
TSO:  
FILENAME PDS 'TSO.PDS';  
FILENAME MEMBER 'TSO.PDS(MEMBER)';
```

%INC

%INCLUDE is a SAS statement that may be used to access and immediately execute additional SAS statements. Other than how to initiate a SAS Display Manager Session, the non-programmer need only know the syntax for the **%INC** statement to interact with data through a system of macro windows. By default, DATA, PROC and macro statements that are executed through use of **%INC** are not entered into the SAS LOG. This lack of log entry is attractive when executing fully tested statement blocks or systems. **%INC** is a host dependent statement.

DOS:
%INC PROGRAM;

Windows:
%INC PROGRAM;
%INC DIRECT(PROGRAM.PGM);

TSO
%INC MEMBER;
%INC PDS(MEMBER);

FUNCTION KEYS

Individual or multiple Display Manager commands and macro invocation may be assigned to function keys. DATA Step, PROC and macro statements may be submitted by entry of a single function key. Function key assignments offer significant ease of command or system execution. However, the complexity of key combinations, the differences of outcome by platform and the ability to modify default assignments lead me to conclude that the **%INC** statement is better suited to the less sophisticated computer user.

The Display Manager provides for function key assignment through the KEYS window, the **KEYDEF** command or the **%KEYDEF** statement. Permanent assignment may be made through the KEYS window while assignments by command are limited to the duration of the SAS session or until changed.

MACRO AUTOCALL

An additional alternative is to use the Macro Autocall facility. Use of this facility requires the MAUTOSOURCE system option to be active and the SASAUTO= system option must allocate the storage location for macro members. The systems options of the Macro Autocall facility may be conveniently set by inclusion in an Autoexec file.

STORED COMPILED MACRO FACILITY

The Stored Compiled Macro Facility was introduced in the SAS 6.07 release. Compiled Macros populate a SAS CATALOG(s) within a SAS Library(s). Consideration of the advantages and disadvantages of this facility are not addressed here. However, in some manner the storage location of the macros must still be allocated. Autoexec allocation causes the method and syntax of allocation to be transparent to the non-programming user.

MACRO

Included as part of Base SAS, the macro facility provides for the packaging of text into units that, when called by name, can be especially useful in reducing the volume of SAS language statement that are required to accomplish repetitive or modular processing. A macro definition begins and end with the macro program statements **%MACRO** and **%MEND** respectively. Once compiled, the macro may be called anytime for the duration of the SAS session.

%MACRO EXAMPLE;
%MEND EXAMPLE;
%EXAMPLE

The **%** causes the macro processor to act upon a macro program statement or a macro function. Macro programming statements that define a macro or macro window or modify the global environment can be entered interactively, exist in open code, or be resident in a macro definition. Statements of processing logic and the **%LOCAL** statement are only allowed within a macro definition.

Macro variables may be created by a number of macro program statements or within a DATA Step interface. The **%LET** statement provides the most direct method of assigning a text value to a macro variable.

%LET VAR=TEXT STRING;

A macro variable's referencing environment exists as either global to the SAS session or as local to a macro. **%GLOBAL** and **%LOCAL** are programming statements that allow for specific assignment of a variable to a referencing environment. However, parameters are always local to their macro. The **%LOCAL** statement is

the only means for macro variable creation that must exist inside a macro definition.

```
%GLOBAL VAR1 VAR2 VAR3;
```

Macro character functions are applied to text strings with similar results as like named SAS language functions, however, the range of macro facility functions is significantly limited. Results are quoted when using some character functions.

```
%IF %LENGTH(&VAR)=7 %THEN  
  %IF %UPCASE(%SUBSTR(&VAR,5,2))=XX %THEN %DO;
```

The %EVAL function evaluates text strings as logical expressions, or performs integer calculations on text strings that are resolved as having integer properties. The distinction between an explicit and an implicit use of %EVAL is not covered here but will be of interest to the advanced macro developer.

```
%LET NUM=%EVAL(&VAR*100);
```

DATA step interfaces of SYMGET and SYMPUT provide for the assignment or the employment of a macro variable's value within an iteration of a DATA step. Being very resource expensive, use of DATA step interfaces should be minimized.

```
IF _ERROR_ NE 0 THEN DO;  
  CALL SYMPUT('WHY','INVALID PROCESSED START DATE');  
  CALL SYMPUT('NOGO','YES'); END;
```

Use of & invokes the macro processor to resolve a macro variable reference. Assuming the existence of a symbolic variable named VAR, the macro processor will substitute the text string value of the symbolic variable each time the &VAR statement is encountered.

Compound & use causes the macro processor to scan a word multiple times. When encountered, && will be resolved to & by the macro processor. Should macro variables exist named VAR1 and INDX, then &&VAR&INDX would be scanned twice. Where the text value of INDX is 1, the resolution of the first scan would look to the macro processor as &VAR1. Substitution of the text of the macro variable VAR1 will result from the second scan by the macro processor.

```
%DO I=1 %TO 5;  
  %IF &&VAR&I NE %THEN %DO;
```

%WINDOW

First available in Release 6.06, the %WINDOW statement allows for macro processor controlled custom windows. Macro windows are defined by name and, once defined, are available for the duration of the SAS session. Its name appears at the top of a window when displayed. Function keys, the command line, and the PMENU facility are all supported.

A field of a macro window is definable as text or as a symbolic variable. Definition may include length, windowed position and display color. Fields may be combined into groups that are optionally and individually displayable within the same macro window definition.

Macro variables that are first defined from inside a macro window definition are created within the referencing environment that is current. The value of a symbolic variable may be displayed, or the designer can choose not to display entered characters. A variable field may accept assignment or require its value or be protected. Options for variable definition include autoskip and various display attributes.

Windows are best design for the most restrictive display environment. Use generous amounts of empty space and blank lines to avoid widow clutter. Introduce only concise and specific text. Color and highlight sparingly and strategically to enhance the informational value of text. As part of the windows contents, inform the user how to respond either to proceed or to exit without action. Maintain design consistency in a system of windows.

The %WINDOW statement may be used within a macro, in an interactive SAS session, or in open code. However, a window that has been defined internally will be redefined with each execution of the macro. A macro window that will display multiple time, using an unaltered definition, will be more efficient if executed outside a repetitively executing macro. Endeavor to place macro window definitions in open code or the outermost level of macro nesting.

```
%WINDOW NOGO  
#01 @14 "MEDICARE DATA - SELECTION ERROR &SYSDATE"  
      COLOR=CYAN  
#06 @14 WHY &0 COLOR=YELLOW PROTECT=YES  
#11 @ 14 "PRESS <ENTER> TO CONTINUE";
```

%DISPLAY

A windowing environment is managed by the SAS system for the display of macro windows. This macro windowing environment is separate from both the display manager windowing and the macro variable referencing environments.

Defined macro windows are displayable using the %DISPLAY statement. Used outside a macro, each execution of a %DISPLAY statement creates a windowing environment that is unique to and exists only for the duration of an individual window. The first %DISPLAY statement used inside a macro creates the windowing environment that is then used by any additional %DISPLAY statements contained in that same macro. Placement of all %DISPLAY statements within the same macro will enhance a windowing system's efficiency.

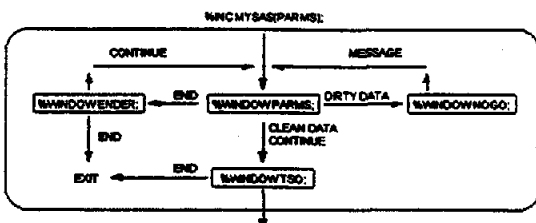
```
%MACRO EXAMPLE;
  %DISPLAY PARM;
  %IF &NOGO=YES %THEN %DISPLAY NOGO;
  %ELSE %DISPLAY TSO; %END;
%MEND EXAMPLE;
%EXAMPLE;
```

%INC SELECT;

For the example system, the library that contains the member SELECT was allocated by inclusion of a LIBNAME statement in an Autoexec file. Use of %INC results in the member being included and immediately submitted. SAS language and macro statements that constitute the member will not be listed in the SAS log.

The purpose of SELECT is to create a SAS data set with a single observation with a number of variables. Each variable will act to control the processing of remote data files. SELECT offers the user a system of macro windows for the entry and editing of symbolic variables. The edited texts of the macro variables become data values for SET variables. Included in the member are window definitions, a macro definition and DM statements.

%MACRO PARMS;



%MACRO PARMS controls the display of a system of four macro windows. The PARMS window will display macro variable values and accepts keyboard entry. Error messages, that result from edit logic applied to the variables of the PARMS window, are displayed in the window named NOGO. A window named ENDER is displayed as a way to exit the system. Display of a window named TSO reminds the system user that the remote session must be at the TSO READY prompt. The TSO window displays only if there are no PARMS variables in error.

```
%MACRO PARMS;
%DO %UNTIL(&NOGO NE YES);
  %DISPLAY PARM;

  %LET NOGO=;
  %LET WHY=;
  %IF &FILE1 NE %THEN %DO;
    DATA _NULL_;
    CHAR="&FILE1"||"01";
    CHECK=INPUT(CHAR,YYMMDD6);
    IF _ERROR_ NE 0 THEN DO;
      CALL SYMPUT('WHY','INVALID PROCESSED START DATE');
      CALL SYMPUT('NOGO','YES'); END;
    RUN;

    %IF &NOGO NE YES %THEN %DO;
      DATA PARM;
      FILE1=&FILE1;
      %DO I=1 %TO 5;
        %IF &&PRID&I NE %THEN %DO;
          PRID&I="||||"||"&PRID&I"||"||"; %END;
        %ELSE %DO;
          PRID&I=" "; %END; %END;
      RUN; %END; %END;

    %ELSE %DO;
      %DISPLAY ENDER;
      %LET NOGO=YES;
      %LET WHY=INVALID PROCESSED START DATE; %END;

    %IF &NOGO=YES %THEN %DISPLAY NOGO;
    %ELSE %DISPLAY TSO; %END;

  %MEND PARM;
%PARMS;
```

DM

SAS Display Manager offers some 100 commands for navigating the course of an interactive SAS session. With the DM SAS language statement, Display Manager commands may be submitted as SAS language statements. Imbedded DM statements at the end of the SELECT file will automate a SAS/CONNECT signon, include a macro definition and submit the macro for processing on a remote host.

It is syntactically correct to omit the semicolon from a macro call statement, however, to do so prior to a DM statement seems to confuse the system. Semicolon inclusion is both correct in syntax and will eliminate command confusion.

```
%PARMS;
DM 'SIGNON';
DM 'INC 'CAANALYSIS' TOPPROG BIGBOX.MAC';
DM 'RSUBMIT';
```

SAS/CONNECT

Data may be resident on multiple platforms with diverse operating environments. SAS/CONNECT offers the opportunity to span platforms and link environments. Files and DATA Sets, catalogs and, with SAS 6.08, entire libraries may be transferred among the links. Program logic may be submitted for processing on a remote host with the results returned to the local host.

%MACRO BIGBOX:

BIGBOX is a macro intended to run on the remote host. The file may be resident on the remote host or may be submitted remotely from the local host. In this instance, a gray area of corporate standards places the file on the local host.

```
%MACRO BIGBOX;
```

A LIBNAME statement is used to allocate user specific storage for the accommodation of temporary data. Space allocated to the WORK library remains available to SAS utilities.

```
LIBNAME USER '&TEMP' SPACE=(TRK,(1000,100))
UNIT=SYSDA;
```

An upload of data from the local host is input to a _NULL_ Data Step. Uploaded variables are used to create macro variables through the SYMPUT DATA Step interface.

```
PROC UPLOAD DATA=PARMS OUT=PARMS;
RUN;
```

```
DATA _NULL_;
SET PARMS;
CALL SYMPUT('GOAL','GOAL');
%DO I=1 %TO 2;
CALL SYMPUT('FILE&I','FILE&I'); %END;
CALL SYMPUT('FROM','FROM');
CALL SYMPUT('THRU','THRU');
%DO I=1 %TO 5;
CALL SYMPUT('PRID&I','PRID&I'); %END;
RUN;
```

Substitution of macro text completes the SAS language statements that produce of a View of an external data file. The conventions of having the names of monthly data files end in YYMM allow for the reading of multiple data files, as specified by macro resolution. Processing may be iterative if multiple data files are to be read.

```
%DO A=1 %TO 2;
%IF &A FROM &A NE %THEN
%DO B=&A FROM &A %TO &A THRU &A
FILENAME READ "PROD.DAT&B";
DATA SELECT / VIEW=SELECT;
INFILE READ MISSEVER;
INPUT @&423 FROM ?? YYMMDD6;
@&429 THRU ?? YYMMDD6; @
IF FROM GE &FROM AND THRU LE &THRU;
INPUT @&421 PRID S07;
%IF &PRID1 NE %THEN %DO;
IF PRID IN (&PRID1 &PRID2 &PRID3 &PRID4 &PRID5);
%END;
RUN;
```

The remote host is commanded to download each data set, as created directly from the Data View, to the local host. No intermittent data set(s) is constructed on the remote host. Macro substitution allows for dynamic naming of the local host data set. Signoff is accomplished using the DM statement.

```
PROC DOWNLOAD DATA=SELECT OUT=LOCAL.&GOAL&B;
%END; %END;
```

```
%MEND BIGBOX;
%BIGBOX;
```

```
ENDRSUBMIT;
DM 'SIGNOFF';
```

CONCLUSIONS:

While the SAS software system offers an array of tools for application development and data analysis, a site may be limited as to the products installed. There is sufficient sophistication in Base SAS for the development of user-friendly access and analysis systems for the non-programming computer user.

The developer should strive to make the methods for allocation transparent and system activation friendly. File allocation from inside an AUTOEXEC file and use of the %INC statement for system activation combine as one of a number of alternatives. Weigh the positive and negative aspect of each alternative relative to the application and the user. Implement a naming convention that will allow same record layout data files, that are individual according to a convenience criterion, to be accessed repetitively through the resolution of symbolic variables.

%WINDOW statements are best placed in open code or in the highest level of macro nesting possible. Design window with large amounts of blank space and with consideration for the most restrictive display environment. Make text brief and exact. Specify methods for response with, or exit without, action. Be conservative in color

and highlighting. Strive to place all %DISPLAY statements inside the same macro. Give windowed systems a consistent look.

Where the macro invocation is followed by a DM statement, delimit the invocation with a semicolon. Submit Display Manager commands using the DM statement to automate remote signon, submit and signoff. As the values of macro variables cannot span environments, upload a set of data variables as populated on the local host from the resolution of macro

variables. Use the uploaded data set to create remote host macro variables with the SYMPUT DATA Step interface. Reduce the negative impact on system resources by limiting the use of macro DATA Step interfaces.

When downloading data that is not in the SAS format, use a VIEW to avoid the creation of an intermittent data set on the remote host. Be aware that the movement of large amounts of data will be slow and may have a negative effect on other computer users.

SAS For Windows 6.08
%DISPLAY PARMS;

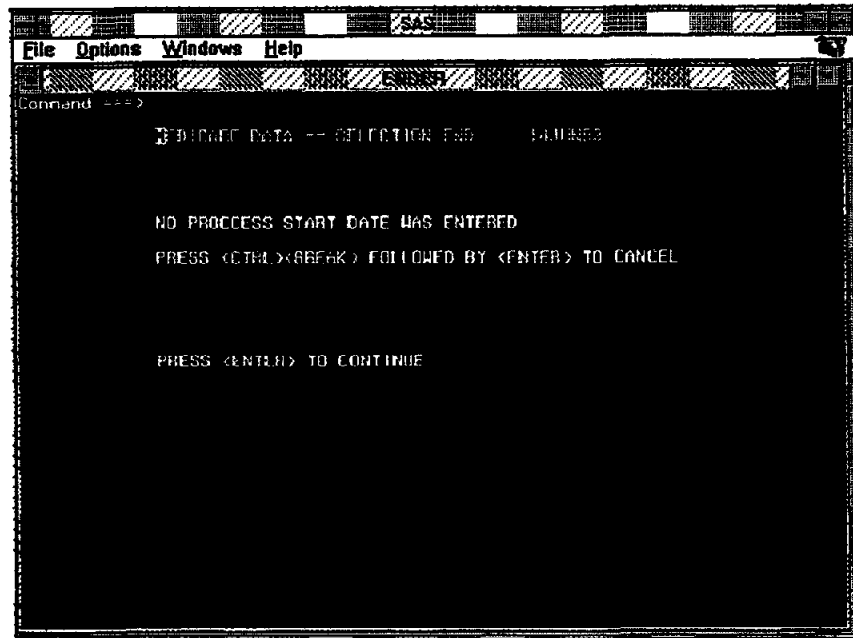
```

File Options Windows Help
PARMS
Command ---> |
DETERMINE DATA -- SELECTION & DATA FILE -- (REQUIRED)
ENTER SELECTION CRITERIA -- LEAVE BLANK FOR ALL
* UP TO FIVE ENTRIES ACCEPTED
PREFIX OUTPUT DATASET
PROCESSED YEAR START      _____  END      _____
SERVICE YEAR FROM      _____  THRU      _____
* PROVIDER NUMBER      _____
BILL TYPE (2 POS)      _____  (3RD POS)  _____
* PRIMARY DIAGNOSIS
* HCPCS CODES
* REVENUE CODE
PRESS <ENTER> TO CONTINUE      LEAVE START AS UNDERSCORES TO EXIT
  
```

SAS and SAS/CONNECT are registered trademarks of SAS Institute Inc. in the USA and other countries.

Jim Anderson
Senior Systems Analyst
Blue Cross Blue Shield of Nebraska
7261 Mercy Road
PO Box 3248
Omaha, Nebraska 68180-0001
402-390-1800 Ex. 1310

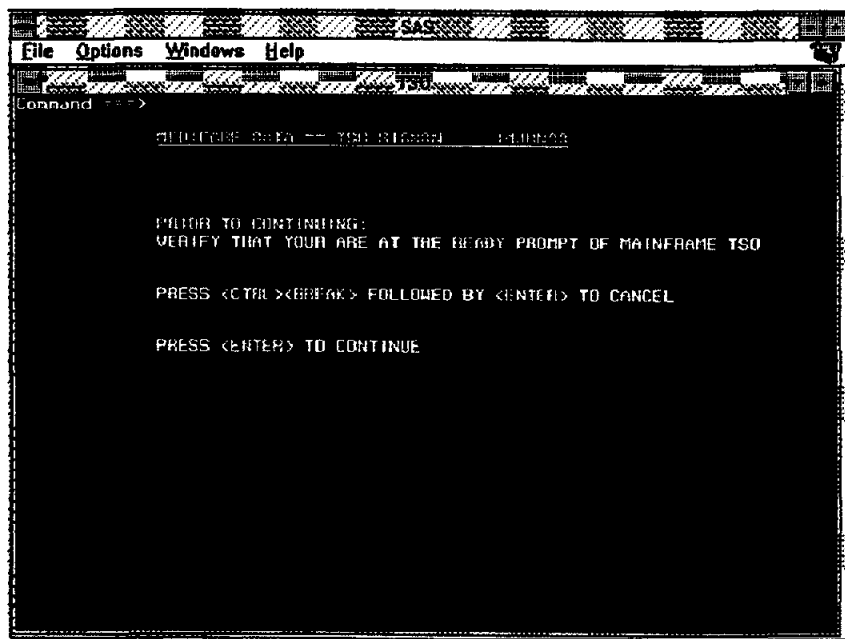
%DISPLAY ENDER;



%DISPLAY NOGO;



%DISPLAY TSO;



LIBNAME TOP 'C:\ANALYSIS\STOPMR';

%GLOBAL WHY GOAL FILE1 FILE2 START END FROM THRU
PRID1 PRID2 PRID3 PRID4 PRID5 BILL BILL3
DIAG1 DIAG2 DIAG3 DIAG4 DIAG5
HCPCS1 HCPCS2 HCPCS3 HCPCS4 HCPCS5
REVCODE1 REVCODE2 REVCODE3 REVCODE4 REVCODE5;

%WINDOW PARMS

#01 @14 "MEDICARE DATA - SELECTION & DOWNLOAD &SYSDATE"
COLOR=CYN
#02 @14 "ENTER SELECTION CRITERIA - LEAVE BLANK FOR ALL"
#03 @12 "" COLOR=YELLOW
@14 "UP TO FIVE ENTRIES ACCEPTED"
#05 @14 "PREFIX"
@21 "OUTPUT" COLOR=YELLOW
@28 "DATASET"
@40 GOAL 4 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
#07 @14 "PROCESSED"
@24 "YMM" COLOR=YELLOW
@29 "START"
@40 FILE1 4 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@30 "END"
@56 FILE2 4 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
#09 @14 "SERVICE"
@22 "YMMDD" COLOR=YELLOW
@30 "FROM"
@40 FROM 6 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@50 "THRU"
@56 THRU 6 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
#11 @12 "" COLOR=YELLOW
@14 "PROVIDER NUMBER"
@40 PRID1 4 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@45 PRID2 4 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@50 PRID3 4 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@55 PRID4 4 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@60 PRID5 4 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
#13 @14 "BILL TYPE (2 POS)"
@40 BILL 2 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@50 "3RD POS"
@62 BILL3 1 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
#15 @12 "" COLOR=YELLOW
@14 "PRIMARY DIAGNOSIS"
@40 DIAG1 5 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@46 DIAG2 5 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@52 DIAG3 5 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@58 DIAG4 5 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@64 DIAG5 5 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
#17 @12 "" COLOR=YELLOW
@14 "HCPCS CODES"
@40 HCPCS1 5 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@46 HCPCS2 5 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@52 HCPCS3 5 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@58 HCPCS4 5 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@64 HCPCS5 5 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
#19 @12 "" COLOR=YELLOW
@14 "REVENUE CODE"
@40 REVCODE1 3 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@44 REVCODE2 3 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@48 REVCODE3 3 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@52 REVCODE4 3 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
@56 REVCODE5 3 AUTOSKIP=YES COLOR=CYN ATTR=UNDERLINE
#21 @08 "PRESS"
@14 "<ENTER>" COLOR=YELLOW
@22 "TO CONTINUE LEAVE"
@44 "START" COLOR=YELLOW
@50 "AS UNDERScores TO EXIT";

%WINDOW NOGO

#01 @14 "MEDICARE DATA - SELECTION ERROR &SYSDATE"
COLOR=CYN
#06 @14 WHY 60 COLOR=YELLOW PROTECT=YES
#11 @14 "PRESS"
#11 @20 "<ENTER>" COLOR=YELLOW
#11 @28 "TO CONTINUE";

%WINDOW ENDER

#01 @14 "MEDICARE DATA - SELECTION END &SYSDATE"
COLOR=CYN
#06 @14 "NO PROCESS START DATE WAS ENTERED"
#08 @14 "PRESS"
#08 @20 "<CTRL><BREAK>" COLOR=YELLOW
#08 @34 "FOLLOWED BY"
#08 @46 "<ENTER>" COLOR=YELLOW
#08 @54 "TO CANCEL"
#14 @14 "PRESS"
#14 @20 "<ENTER>" COLOR=YELLOW
#14 @28 "TO CONTINUE";

%WINDOW TSO

#01 @14 "MEDICARE DATA - TSO SIGNON &SYSDATE"
COLOR=CYN
#06 @14 "PRIOR TO CONTINUING:" COLOR=YELLOW
#07 @14 "VERIFY THAT YOUR ARE AT THE"
#07 @42 "READY" COLOR=YELLOW
#07 @48 "PROMPT OF MAINFRAME TSO"
#10 @14 "PRESS"
#10 @20 "<CTRL><BREAK>" COLOR=YELLOW
#10 @34 "FOLLOWED BY"
#10 @46 "<ENTER>" COLOR=YELLOW
#10 @54 "TO CANCEL"
#13 @14 "PRESS"
#13 @20 "<ENTER>" COLOR=YELLOW
#13 @28 "TO CONTINUE";

%MACRO PARMS;

%DO %UNTIL(ANOGO NE YES);
%DISPLAY PARMS;

%LET NOGO=;
%LET WHY=;
%IF &FILE1 NE %THEN %DO;
DATA _NULL_;
CHAR="&FILE1"||"01";
CHECK=INPUT(CHAR,YMMDD6.);
IF _ERROR_ NE 0 THEN DO;
CALL SYMPUT(WHY,INVALID PROCESSED START DATE);
CALL SYMPUT(NOGO,YES); END;
RUN;

%IF &NOGO NE YES %THEN %DO;
%IF &FILE2 NE %THEN %DO;
DATA _NULL_;
CHAR="&FILE2"||"01";
CHECK=INPUT(CHAR,YMMDD6.);
IF _ERROR_ NE 0 THEN DO;
CALL SYMPUT(WHY,INVALID PROCESSED END DATE);
CALL SYMPUT(NOGO,YES); END;
RUN; %END; %END;

%IF &NOGO NE YES %THEN %DO;
DATA PARMS;
FILE1=&FILE1;

%IF &GOAL NE %THEN %DO;
GOAL="&GOAL"; %END;
%ELSE %DO;
GOAL="SET"; %END;

%IF &NOGO NE YES %THEN %DO;
%IF &FROM NE %THEN %DO;
CHAR="&FROM";
FROM=INPUT(CHAR,YMMDD6.);
IF _ERROR_ NE 0 THEN DO;
CALL SYMPUT(WHY,INVALID SERVICE FROM DATE);
CALL SYMPUT(NOGO,YES); END;
DROP CHAR; %END;
%ELSE %DO;
FROM=""; %END; %END;

%IF &NOGO NE YES %THEN %DO;
%IF &THRU NE %THEN %DO;
CHAR="&THRU";
THRU=INPUT(CHAR,YMMDD6.);
IF _ERROR_ NE 0 THEN DO;
CALL SYMPUT(WHY,INVALID SERVICE THRU DATE);
CALL SYMPUT(NOGO,YES); END;
DROP CHAR; %END;
%ELSE %DO;
THRU=""; %END; %END;

%DO I=1 %TO 5;
%IF &PRID&I NE %THEN %DO;
PRID&I="||"&PRID&I||""; %END;
%ELSE %DO;
PRID&I=" "; %END; %END;

%IF &BILL NE %THEN %DO;
BILL="||"&BILL||""; %END;
%ELSE %DO;
BILL=" "; %END;

%IF &BILL3 NE %THEN %DO;
BILL3="||"&BILL3||""; %END;
%ELSE %DO;
BILL3=" "; %END;

%IF &NOGO NE YES %THEN %DO;
%DO I=1 %TO 5;
%IF &DIAG&I NE %THEN %DO;

```

%IF %LENGTH(&&DIAG&I) LT 3 %THEN %DO;
%LET WHY=INVALID DIAGNOSIS CODE;
%LET NOGO=YES; %END;
%ELSE %DO;
DIAG&I="I"&&DIAG&I"II"; %END; %END;
%ELSE %DO;
DIAG&I=" "; %END; %END; %END;

%IF &NOGO NE YES %THEN %DO;
%DO I=1 %TO 5;
%IF &&HCPCS&I NE %THEN %DO;
%IF %LENGTH(&&HCPCS&I) LT 3 %THEN %DO;
%LET WHY=INVALID HCPCS CODE;
%LET NOGO=YES; %END;
%ELSE %DO;
HCPCS&I="I"&&HCPCS&I"II"; %END; %END;
%ELSE %DO;
HCPCS&I=" "; %END; %END; %END;

%IF &NOGO NE YES %THEN %DO;
%DO I=1 %TO 5;
%IF &&REVCODE&I NE %THEN %DO;
%IF %LENGTH(&&REVCODE&I) LT 3 %THEN %DO;
%LET WHY=INVALID REVENUE CODE;
%LET NOGO=YES; %END;
%ELSE %DO;
REVCODE&I="I"&&REVCODE&I"II"; %END; %END;
%ELSE %DO;
REVCODE&I=" "; %END; %END; %END;

RUN; %END; %END;

%ELSE %DO;
%DISPLAY ENDER;
%LET NOGO=YES;
%LET WHY=INVALID PROCESSED START DATE;
%LET NOGO=YES; %END;

%IF &NOGO=YES %THEN %DISPLAY NOGO;
%ELSE %DISPLAY TSO; %END;

%MEND PARMS;
%PARMS;

DM 'SIGNON';
DM 'INC H:USERS\HSR\SAS\MEDICARE\BIGBOX.MAC';
DM 'SUBMIT';

%MACRO BIGBOX;

LIBNAME USER &TEMP SPACE=(TRK,(4500,450)) UNIT=SYSDA;
PROC UPLOAD DATA=PARMS OUT=PARMS;
RUN;

DATA _NULL_;
SET PARMS;
CALL SYMPUT('GOAL',GOAL);
%DO I=1 %TO 2;
CALL SYMPUT('FILE&I',FILE&I); %END;
CALL SYMPUT('FROM',FROM);
CALL SYMPUT('THRU',THRU);
%DO I=1 %TO 5;
CALL SYMPUT('PRID&I',PRID&I); %END;
CALL SYMPUT('BILL',BILL);
CALL SYMPUT('BILL3',BILL3);
%DO I=1 %TO 5;
CALL SYMPUT('DIAG&I',DIAG&I); %END;
%DO I=1 %TO 5;
CALL SYMPUT('HCPCS&I',HCPCS&I); %END;
%DO I=1 %TO 5;
CALL SYMPUT('REVCODE&I',REVCODE&I); %END;
RUN;

%DO A= &FILE1 %TO &FILE2;
FILENAME READ 'F0183.MC&A';
DATA SELECT / VIEW=SELECT;
INFILE READ MISSEVER;
INPUT @0023 FROM ?? YYMMDD6. /* FROM DATE */
@0429 THRU ?? YYMMDD6. @; /* THRU DATE */
%IF &FROM NE %THEN %DO;
IF FROM GE &FROM AND THRU LE &THRU; %END;

INPUT @1818 BILL $02. @; /* BILL TYPE - 2 POS */
%IF &BILL NE %THEN %DO;
IF BILL=&BILL; %END;

INPUT @1820 BILL3 $01. @; /* BILL TYPE - 3RD POS */
%IF &BILL3 NE %THEN %DO;
IF BILL3=&BILL3; %END;

INPUT @0001 HICN $12. /* HICN */
@0021 PRID $07. @; /* PROVIDER NUMBER */
%IF &PRID1 NE %THEN %DO;
IF PRID IN (&PRID1 &PRID2 &PRID3 &PRID4 &PRID5); %END;

INPUT @0103 DCN $13. /* DOC CONTROL NUM */
@0122 DIAG $05. @; /* PRIMARY DISGNOSIS */
%IF &DIAG1 NE %THEN %DO;
%IF %LENGTH(&DIAG1)=7 %THEN
%IF %UPCASE(%SUBSTR(&DIAG1,5,2))=XX %THEN %DO;
%LET PREFIX=%SUBSTR(&DIAG1,2,3);
IF DIAG GE "&PREFIX.00" AND DIAG LE "&PREFIX.99"; %END;
%ELSE %DO;
IF DIAG IN (&DIAG1 &DIAG2 &DIAG3 &DIAG4 &DIAG5); %END;
%ELSE %DO;
IF DIAG IN (&DIAG1 &DIAG2 &DIAG3 &DIAG4 &DIAG5); %END;
%END;

INPUT @0433 DSCHSTAT $01. /* DISCHARGE STATUS */
@0551 MEDREC $17. /* MEDICAL RECORD NUM */
@0628 PAYDT ?? YYMMDD6. /* PAYMENT DATE */
@0634 REIM PD6.2 /* REIMBURSEMENT AMT */
@1386 PTZIP $09. /* PATIENT ZIP CODE+4 */
@1395 PTSEX $01. /* PATIENT SEX */
@1396 PTDOB PD4. /* PATIENT BIRTH DATE */
@1400 CENT $01. /* CENTURY */
@2719 PTCNTR $17. /* PATIENT CONTROL NUM */
@3215 BYTES 03. @; /* REV-CH LINE SIZE */

AT=3223;

DO I=1 TO BYTES;
LNENO=I;
KEEPER='Y';
INPUT @AT HCPCS $05. /* REV HCPC4 */
MOD1 $02. /* REV HCPC MOD1 */
+6 REVCODE $03. /* REV CODE */
+1 UNIT PD4. /* REV UNITS */
UNITDEN PD4. /* REV UNITS DENIED */
+5 CHCOV PD6.2 /* REV TOT COV CHRGS */
CHNONCOV PD6.2 /* REV TOT NCOV CHRGS */
CHOTHER PD6.2 @; /* REV TOT OTH CHRGS */

%IF &HCPCS1 NE %THEN %DO;
%IF %LENGTH(&HCPCS1)=7 %THEN %DO;
%IF %UPCASE(%SUBSTR(&HCPCS1,5,2))=XX %THEN %DO;
%LET PREFIX=%SUBSTR(&HCPCS1,2,3);
IF HCPCS GE "&PREFIX.00" AND HCPCS LE "&PREFIX.99" THEN;
ELSE KEEPER='N'; %END;
%ELSE %DO;
IF HCPCS IN (&HCPCS1 &HCPCS2 &HCPCS3 &HCPCS4 &HCPCS5)
THEN;
ELSE KEEPER='N'; %END; %END; %END;

%IF &REVCODE1 NE %THEN %DO;
%IF %LENGTH(&REVCODE1)=5 %THEN
%IF %UPCASE(%SUBSTR(&REVCODE1,4,1))=X %THEN %DO;
%LET PREFIX=%SUBSTR(&REVCODE1,2,2);
IF REVCODE GE "&PREFIX.0" AND REVCODE LE "&PREFIX.9"
THEN;
ELSE KEEPER='N'; %END;
%ELSE %DO;
IF REVCODE IN (&REVCODE1 &REVCODE2 &REVCODE3
&REVCODE4
&REVCODE5) THEN;
ELSE KEEPER='N'; %END;
%ELSE %DO;
IF REVCODE IN (&REVCODE1 &REVCODE2 &REVCODE3
&REVCODE4
&REVCODE5) THEN OUTPUT;
ELSE KEEPER='N'; %END; %END;

IF KEEPER='Y' THEN OUTPUT;

AT=AT+73; END;
DROP AT BYTES I KEEPER;
RUN;
PROC DOWNLOAD DATA=SELECT OUT=TOP.&GOAL&A;
RUN; %END;

%MEND BIGBOX;
%BIGBOX;

ENDRSUBMIT;
DM 'SIGNOFF';

```