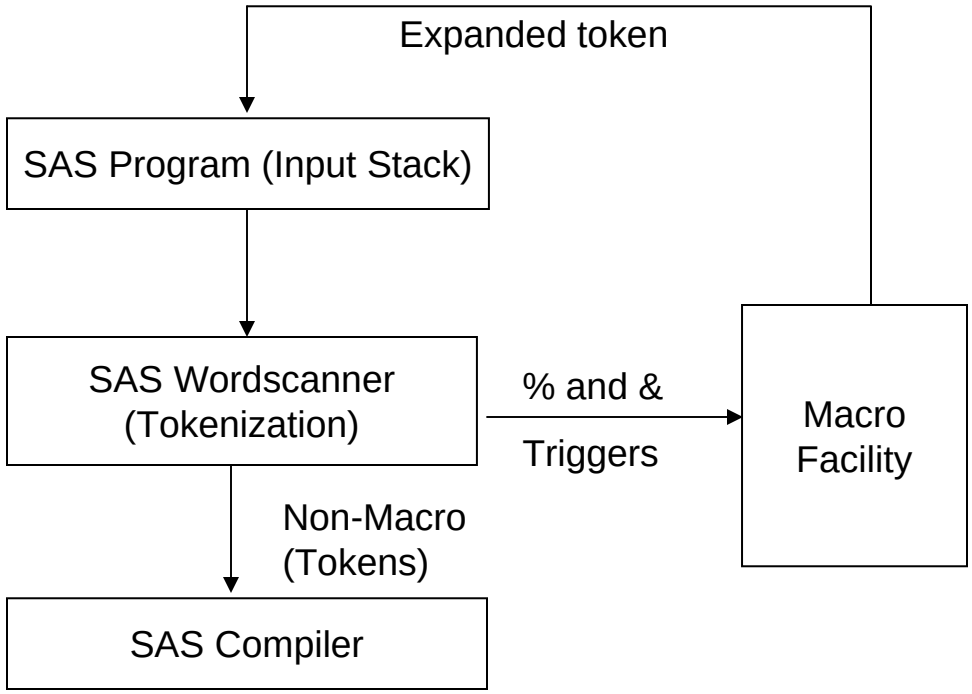
An Introduction to SAS® Macros



SYSTEMS SEMINAR CONSULTANTS, INC.

Steven First, President
2997 Yarmouth Greenway Drive, Madison, WI 53711
Phone: (608) 278-9964, Web: www.sys-seminar.com

An Introduction to SAS® Macros

This course was written by Systems Seminar Consultants, Inc. SSC specializes SAS software and offers SAS:

- Training Services
- Consulting Services
- Help Desk Plans
- Newsletter subscriptions to *The Missing Semicolon™.*

Objectives

After completing this class students will:

- understand how the macro system fits with the rest of SAS software
- use system (automatic) macro variables
- create and use user defined macro variables
- define simple macros
- pass data from the data step to the macro system.

SAS Macro Overview

Macros construct input for the SAS compiler.

Functions of the SAS macro processor:

- pass symbolic values between SAS statements and steps
- establish default symbolic values
- conditionally execute SAS steps
- invoke very long, complex code in a quick, short way.

Notes:

- The MACRO PROCESSOR is the SAS system module that processes macros.
- The MACRO LANGUAGE is how you communicate with the processor.

Traditional SAS Programming

Without macros what are some things you *cannot* easily do?

- substitute text in statements like TITLEs
- communicate across SAS steps
- establish default values
- conditionally execute SAS steps
- hide complex code that can be invoked easily.

Flow Without Macros

Without macros, SAS programs are DATA and PROC steps.

- The program is scanned one statement at a time looking for the beginning of step (step boundary).
- When the beginning of step is found, all statements in the step are **compiled**.
- When the end of step is found (the next step boundary), the previous step **executes**.

SAS Step Boundaries

Step boundaries are the SAS keywords:

DATA	ENDSAS
PROC	LINES
CARDS	LINES4
CARDS4	PARMCARDS
DATALINES	QUIT
DATALINES4	RUN

Step Boundaries

RUN acts as an explicit step boundary in most PROCs.

```
data saleexps;                  <--Step, start compile
 infile rawin;
 input name $1-10 division $12
       years 15-16 sales 19-25
       expense 27-34;
 run;                           <--Step end, exec previous

proc print data=saleexps;       <--Step start, start compile
 run;                           <--Step end, exec previous
proc means data=saleexps;
 var sales expense;
 run;                           <--Step end, exec previous
```

Notes:

- The use of RUN after each step is highly recommended.

Practice Exercises

Exercise 1:

- A program that will build several datasets used later in this course is provided on your system.
- Start SAS for this workshop WS148.
- Check the SAS log to insure that the program ran correctly, use the LIBNAME command, file icon, or submit:

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

  to verify that the datasets were built correctly.
- Explore SAS if you would like to.

The SAS Macro Language

A second SAS programming language for string manipulation.

Characteristics:

- strings are sequences of characters
- all input to the macro language is a string
- usually strings are SAS code, but don't need to be
- the macro processor manipulates strings and may send them back for scanning.

Macro Language Components

The macro language has several kinds of components.

Macro variables:

- are used to store and manipulate character strings
- follow SAS naming rules
- are NOT the same as DATA step variables
- are stored in memory in a macro symbol table.

Macro statements:

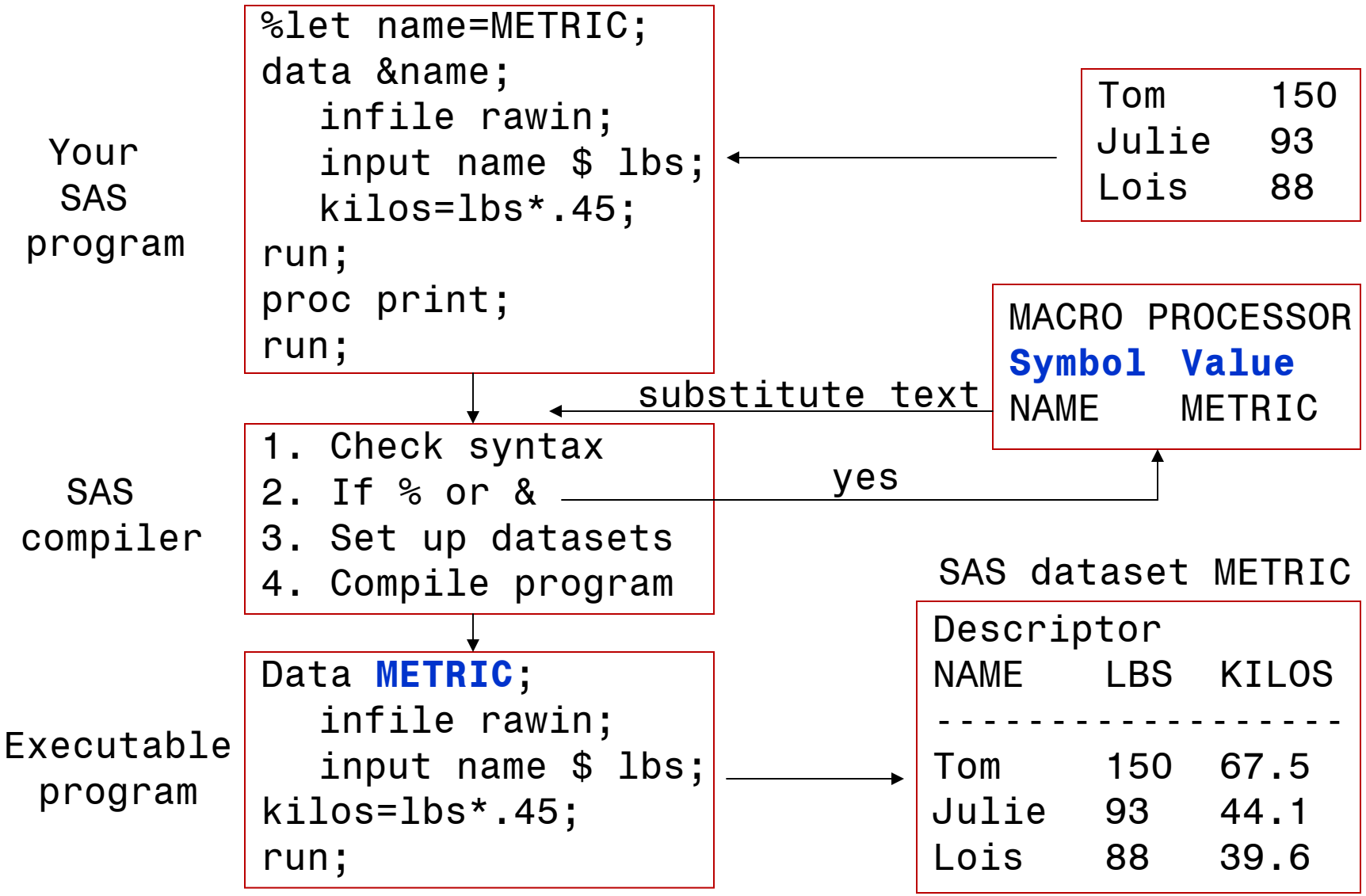
- begin with a % and a macro keyword and end with semicolon (;)
- assign values, substitute values, and change macro variables
- can branch or generate SAS statements conditionally.

Macro Processor Flow

Macro statements are given to the macro processor BEFORE the compiler.

- Macro statements start with %.
- Macro variables are referred to with &.

Macro Processor Flow Example



A Macro Problem

Problem:

- You would like to have the Day of Week and current date appear in a title, but SAS titles are text, not variables.

Solution:

- use some system macro variables.

Solution to Macro Problem

```
PROC PRINT DATA=DEPTSALE;
 TITLE  "Department Sales as of &SYSDAY  &SYSDATE";
 TITLE2 "Deliver to Michael O'Malley";
RUN;
```

Department Sales as of **Wednesday 24MAR99**
Deliver to Michael O'Malley

OBS	DEPT	YEAR	SALARIES	SALES
1	1	87	1000	1000
2	1	88	1000	2000
3	2	87	500	3000
4	2	88	600	2000

Notes:

- Macro variables are NOT resolved within single quotes.

Automatic Macro Variables

A partial list of automatic macro variables and their usage:

```
SYSBUFFR        text entered in response to %INPUT
SYSCMD          last non-SAS command entered
SYSDATE         current date in DATE6. or DATE7. format
SYSDAY          current day of the week
SYSDEVIC        current graphics device
SYSDSN          last SAS dataset built (i.e.,WORK SOFTSALE)
SYSENV          SAS environment (FORE or BACK)
SYSERR          return code set by SAS procedures
SYSFILRC        whether last FILENAME executed correctly
SYSINDEX        number of macros started in job
SYSINFO         system information given by some PROCS
SYSJOBID        name of executing job or user
SYSLAST         last SAS dataset built (i.e., WORK.SOFTSALE)
SYSLIBRC        return code from last LIBNAME statement
SYSLCKRC        whether most recent lock was successful
```

Automatic Macro Variables (continued)

```
SYSMENV         macro execution environment
SYSMSG          message displayed with %DISPLAY
SYSPARM         value passed from SYSPARM in JCL
SYSPROD         indicates whether a SAS product is licensed
SYSPBUFF        all macro parameters passed
SYSRC           return code from macro processor
SYSSCP          operating system where SAS is running
SYSTIME         starting time of job
SYSVER          SAS version
```

Example:

```
FOOTNOTE "THIS REPORT WAS RUN ON &SYSDAY, &SYSDATE";
```

Resolves to:

```
FOOTNOTE "THIS REPORT WAS RUN ON FRIDAY, 26MAR99";
```

Displaying Macro Variables

%PUT displays macro variables to the log at compile time.

Syntax:

```
%PUT text macrovariables ;
%PUT _all_;
```

Example:

```
DATA NEWPAY;
 INFILE DD1;
 INPUT EMP$ RATE;
RUN;
%PUT ***** &SYSDATE *****;
```

Partial SAS Log:

```
***** 26MAR99 *****
```

Displaying All Macro Variables

%PUT can display all current macro variables.

```
%PUT _ALL_;
```

Partial SAS Log:

```
GLOBAL MBILLYR 99
GLOBAL SSCDEV C
AUTOMATIC AFDSID 0
AUTOMATIC AFDSNAME
AUTOMATIC AFLIB
AUTOMATIC AFSTR1
AUTOMATIC AFSTR2
AUTOMATIC FSPBDV
AUTOMATIC SYSBUFFR
AUTOMATIC SYSCMD
AUTOMATIC SYSDATE 26MAR99
AUTOMATIC SYSDAY Tuesday
```

Partial SAS LOG Continued

```
AUTOMATIC SYSDSN          _NULL_
AUTOMATIC SYSENV FORE
AUTOMATIC SYSERR 0
AUTOMATIC SYSFILRC 0
AUTOMATIC SYSINDEX 1
AUTOMATIC SYSINFO 0
AUTOMATIC SYSJOBID 0000016959
AUTOMATIC SYSLAST _NULL_
AUTOMATIC SYSMSG
AUTOMATIC SYSPARM
AUTOMATIC SYSRC 0
AUTOMATIC SYSSCP WIN
AUTOMATIC SYSSCPL WIN_32S
AUTOMATIC SYSSITE 0011485002
AUTOMATIC SYSTIME 10:35
AUTOMATIC SYSVER 6.11
AUTOMATIC SYSVLONG 6.11.0040P030596
```

Practice Exercises

Exercise 2:

- Determine the values of the following system macro variables using your current computer system.

```
&SYSDAY          Current day of the week
&SYSTIME         Current time
&SYSSCP          Operating system being used
&SYSVER          Current SAS version number
&SYSDATE         Current date, in DATE7. Format
```

Exercise 3:

- Using system macro variables run a PROC CONTENTS and a PROC PRINT on the LAST SAS dataset that was created. Include its name in a title.

Exercise 2 Solution

```
%PUT **** SYSDAY  = &SYSDAY;
%PUT **** SYSTIME = &SYSTIME;
%PUT **** SYSSCP  = &SYSSCP;
%PUT **** SYSVER  = &SYSVER;
%PUT **** SYSDATE = &SYSDATE;
```

Exercise 2 Output

```
%PUT **** SYSDAY  = &SYSDAY;
 **** SYSDAY  = Friday
2    %PUT **** SYSTIME = &SYSTIME;
**** SYSTIME = 13:42
3    %PUT **** SYSSCP  = &SYSSCP;
**** SYSSCP  = WIN
4    %PUT **** SYSVER  = &SYSVER;
**** SYSVER  = 6.12
5    %PUT **** SYSDATE = &SYSDATE;
**** SYSDATE = 26MAR99
```

Exercise 3 Solutions

```
proc contents data=&syslast;
 title "contents of &syslast";
run;
```

The Generated Code:

```
proc contents data=WORK.COUNTYDT;
 title "contents of WORK.COUNTYDT";
run;
```

Exercise 3 Partial Output

```
contents of WORK.COUNTYDT
CONTENTS PROCEDURE
Data Set Name:WORK.COUNTYDT                Observations:         6
Member Type:  DATA                         Variables:            2
Engine:       V612                         Indexes:              0
Created:      13:51 Friday, March 26, 1999 Observation Length:   23
Last Modified:13:51 Friday, March 26, 1999 Deleted Observations:0
Protection:                                Compressed:           NO
Data Set Type:                             Sorted:               NO
Label:
```

A Macro Problem

Problem: You reference a SAS datasetname several times in a SAS job.

```
DATA PAYROLL;
 INPUT EMP$ RATE;
 DATALINES;
TOM 10
JIM 10
;
PROC PRINT DATA=PAYROLL;
 TITLE "PRINT OF DATASET PAYROLL";
RUN;
```

Question: How can you change the name quickly in one place only AND have the datasetname appear in a title?

Solution:

- Use a macro variable.

Macro Variables

- You can define macro variables with %LET.
- You refer to the variables later with &variable.
- Macro will substitute value for all occurrences of &variable.

Syntax:

```
%LET variable=value;
```

```
%LET NAME=PAYROLL;
DATA &NAME;
 INPUT EMP$ RATE;
 DATALINES;
TOM 10
JIM 10
;
PROC PRINT DATA=&NAME;
 TITLE "PRINT OF DATASET &NAME";
RUN;
```

The Generated SAS Code

```
DATA PAYROLL;
 INPUT EMP$ RATE;
 DATALINES;
TOM 10
JIM 10
;
PROC PRINT DATA=PAYROLL;
 TITLE "PRINT OF DATASET PAYROLL";
RUN;
```

Notes:

- Macro variables are not resolved within single quotes.
- Leading and trail spaces are discarded.

Assigning a New Value

Use another %LET to assign a different value.

```
%LET NAME=NEWPAY;
DATA &NAME;
 INPUT EMP$ RATE;
 DATALINES;
TOM 10
JIM 10
;
PROC PRINT DATA=&NAME;
 TITLE "PRINT OF DATASET &NAME";
RUN;
```

The Generated SAS Code

```
DATA NEWPAY;
 INPUT EMP$ RATE;
DATALINES;
 TOM 10
 JIM 10
   ;
PROC PRINT DATA=NEWPAY;
TITLE "PRINT OF DATASET NEWPAY";
RUN;
```

Assigning SAS Statements

%STR allows values with ; etc.

```
%LET NAME=NEWPAY;
%LET CHART=%STR(PROC CHART;VBAR EMP;RUN;);
DATA &NAME;
 INPUT EMP$ RATE;
 DATALINES;
TOM 10
JIM 10
;
&CHART
PROC PRINT DATA=&NAME;
 TITLE "PRINT OF DATASET &NAME";
RUN;
```

The Generated SAS Code

system seminar:

```
                    ATE;
 DATALINES;
TOM 10
JIM 10
;
PROC CHART;VBAR EMP;RUN;
PROC PRINT DATA=NEWPAY;
 TITLE "PRINT OF DATASET NEWPAY";
RUN;
```

Nesting of Macro Variables

Macro variables can contain other macro variables.

```
%LET NAME=NEWPAY;
%LET CHART=%STR(PROC CHART DATA=&NAME;VBAR EMP;RUN;);
DATA &NAME;
 INPUT EMP$ RATE;
 DATALINES;
TOM 10
JIM 10
;
&CHART
PROC PRINT DATA=&NAME;
 TITLE "PRINT OF DATASET &NAME";
RUN;
```

The Generated SAS Code

```
DATA NEWPAY;
 INPUT EMP$ RATE;
 DATALINES;
TOM 10
JIM 10
;
PROC CHART DATA=NEWPAY;VBAR EMP;RUN;
PROC PRINT DATA=NEWPAY;
 TITLE "PRINT OF DATASET NEWPAY";
RUN;
```

Practice Exercises

Directions:

- Work out the problems below on paper.
- If terminals are available, log on, type in the statements, and use %PUT statements to check you answers.

Exercise 4:

- After execution of the following %LET statements

```
%LET A=ANDY;
%LET B=1999;
%LET C=CANES;
%LET D=DECEMBER 31,;
%LET E="TREMENDOUS";
```

Exercise 4 (continued)

What would be the results of these %PUT statements?

```
 %PUT &C;

%PUT FISCAL YEAR &B;

%PUT YEAR ENDED &D &B;

%PUT &B C&A&C WERE SOLD IN &B;

%PUT &B WAS A &E SALES YEAR!;
```

Exercise 4 Solution

```
%LET A=ANDY;
%LET B=1999;
%LET C=CANES;
%LET D=DECEMBER 31,;
%LET E="TREMENDOUS";
```

```
          Symbol Table
Name                 Value

A                    ANDY
B                    1999
C                    CANES
D                    DECEMBER 31,
E                    "TREMENDOUS";
```

Exercise 4 Solution

```
%PUT &C;
CANES
%PUT FISCAL YEAR &B;
FISCAL YEAR 1999
%PUT YEAR ENDED &D &B;
YEAR ENDED DECEMBER 31, 1999
%PUT &B C&A&C WERE SOLD IN &B;
1999 CANDYCANES WERE SOLD IN 1999
%PUT &B WAS A &E SALES YEAR!;
1999 WAS A "TREMENDOUS" SALES YEAR!
```

```
          Symbol Table
Name                Value

A                   ANDY
B                   1999
C                   CANES
D                   DECEMBER 31,
E                   "TREMENDOUS";
```

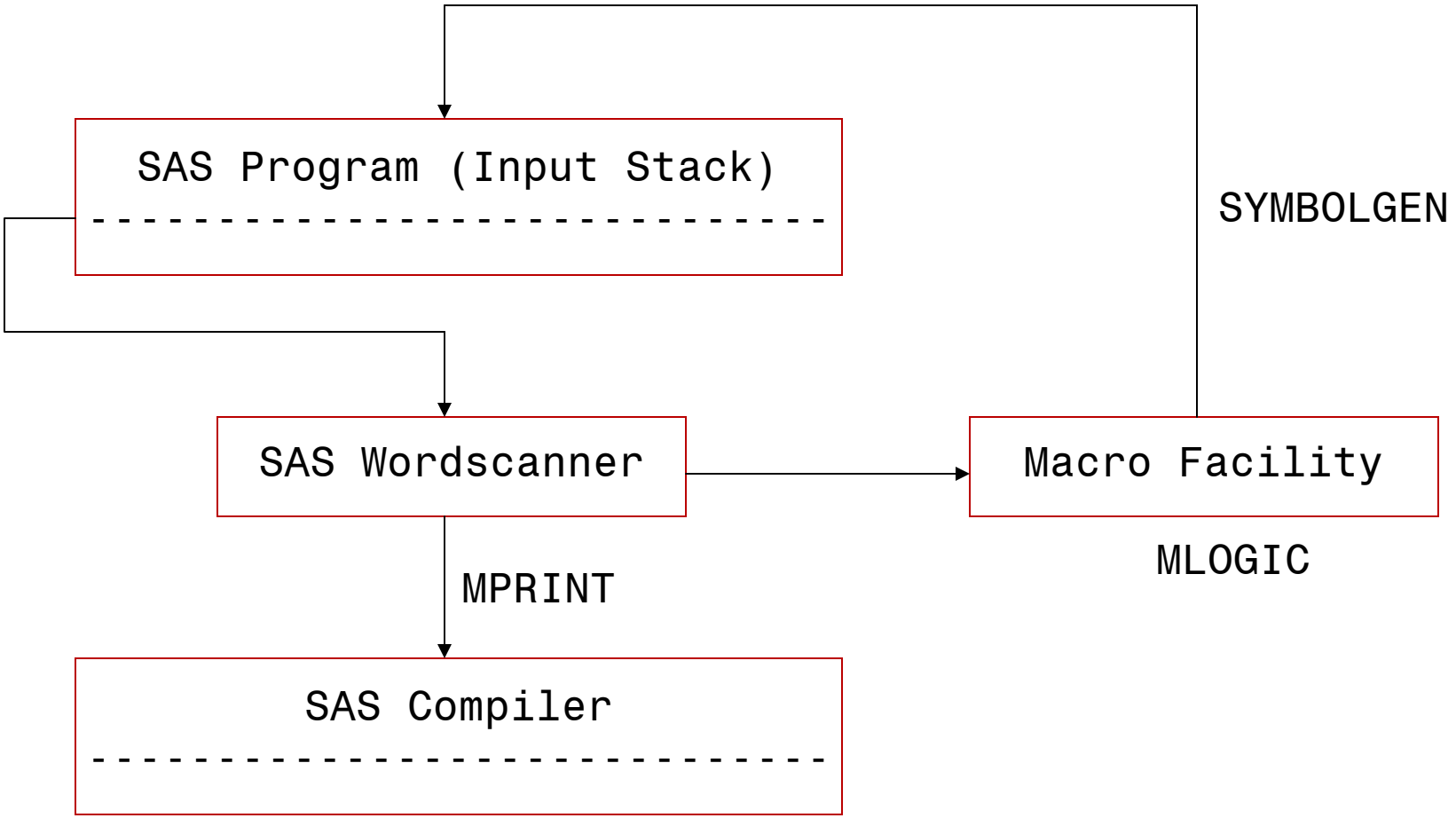
Other Macro Debugging Options

SAS gives several options that control log output.

- SYMBOLGEN/NOSYMBOLGEN controls display of variable resolution
- MPRINT/NOMPRINT displays generated statements given to the compiler
- MLOGIC/NOMLOGIC displays tracing information during macro execution.

Other Macro Debugging Options

A diagram of where macro debugging options display values:



What is a Macro?

Stored text that can be inserted anywhere in a SAS program and expanded.

Macros can include:

- constants such as literals, variables, names, and statements
- assignments to macro variables
- macro programming statements
- macro language functions
- invocations of other functions
- nested macro definitions
- **LOGIC** to conditionally generate SAS code.

Defining and Using Macros

%MACRO and %MEND define macros.

%*macroname* will invoke it later.

Example: Define a macro to run PROC CHART and later invoke

```
%MACRO CHART;
  PROC CHART DATA=&NAME;
    VBAR EMP;
RUN;
%MEND;
```

Invoking the Chart Macro

```
%LET NAME=NEWPAY;
DATA &NAME;
 INPUT EMP$ RATE;
 DATALINES;
TOM 10
JIM 10
;
RUN;
%CHART
PROC PRINT DATA=&NAME;
 TITLE "PRINT OF DATASET &NAME";
RUN;
```

The Generated Code

```
DATA NEWPAY;
 INPUT EMP$ RATE;
 DATALINES;
TOM 10
JIM 10
;
RUN;
PROC CHART DATA=NEWPAY;
 VBAR EMP;
RUN;
PROC PRINT DATA=NEWPAY;
 TITLE "PRINT OF DATASET NEWPAY";
RUN;
```

Practice Exercises

Exercise 5:

- If the following macro was defined to the SAS system:

```
%MACRO FREQ;
    PROC FREQ DATA=&DSN;
      TITLE "EXERCISE 5";
      TABLES &VAR1*&VAR2 / NOPERCENT;
    RUN;
  %MEND FREQ;
```

- What code would the SAS compiler see after these statements?

```
OPTIONS MPRINT SYMBOLGEN;
%LET DSN=FREQ;
%LET VAR1=DEPT;
%LET VAR2=SALES;

%FREQ
```

Exercise 5 Solution

```
PROC FREQ DATA=FREQ;
  TITLE "EXERCISE 5";
  TABLES DEPT*SALES / NOPERCENT;
 RUN;
```

Exercise 5 Output

EXERCISE 5
TABLE OF DEPT BY SALES

DEPT Frequency Row Pct Col Pct	SALES 1000	2000	3000	Total
1	1 50.00 100.00	1 50.00 50.00	0 0.00 0.00	2
2	0 0.00 0.00	1 50.00 50.00	1 50.00 100.00	2
Total	1	2	1	4

Positional Macro Parameters

Macro parameters are defined in order after the macro name.

```
%MACRO CHART(NAME,BARVAR);
  PROC CHART DATA=&NAME;
    VBAR &BARVAR;
  RUN;
%MEND;

%CHART(PAYROLL,EMP)
```

Resolves to:

```
PROC CHART DATA=PAYROLL;
  VBAR EMP;
RUN;
```

Notes:

- Keyword parameters are also allowed, and can set default values.

Nested Macros

Macros can call other macros.

```
%MACRO CHART(NAME,BARVAR);
  PROC CHART DATA=&NAME;
    VBAR &BARVAR;
     RUN;
%MEND;

%MACRO PTCHART(NAME,BARVAR);
%CHART(PAYROLL,EMP)
  PROC PRINT DATA=&NAME;
    TITLE "PRINT OF DATASET &NAME";
  RUN;
%MEND;

%PTCHART(PAYROLL,EMP)
```

The Generated SAS Code

```
PROC CHART DATA=PAYROLL;
 VBAR EMP;
RUN;
PROC PRINT DATA=PAYROLL;
  TITLE  "PRINT OF DATASET PAYROLL";
RUN;
```

Conditional Macro Compilation

%IF can conditionally pass code to the compiler.

Example: Run PROC PRINT only if PRTCH=YES.

```
%MACRO PTCHT(PRTCH,NAME,BARVAR);
  %IF &PRTCH=YES %THEN PROC PRINT DATA=&NAME;
       ;
  PROC CHART DATA=&NAME;
   VBAR &BARVAR;
  RUN;
%MEND;

%PTCHT(YES,PAYROLL,EMP)
```

The Generated SAS Code

```
PROC PRINT DATA=PAYROLL ;

PROC CHART DATA=PAYROLL;
  VBAR EMP;
RUN;
```

The %DO Statement

%DO allows many statements to be conditionally compiled.

Example: Submit as before, but include titles.

```
%MACRO PTCHT(PRTCH,NAME,BARVAR);
  %IF &PRTCH=YES %THEN
      %DO;
          PROC PRINT DATA=&NAME;
          TITLE "PRINT OF DATASET &NAME";
          RUN;
      END;
  PROC CHART DATA=&NAME;
   VBAR &BARVAR;
  RUN;
%MEND;

%PTCHT(YES,PAYROLL,EMP)
```

The Generated SAS Code

```
PROC PRINT DATA=PAYROLL;
 TITLE  "PRINT OF DATASET PAYROLL";
RUN;
PROC CHART DATA=PAYROLL;
 VBAR EMP;
RUN;
```

Interactive Macro Invocation

%DO can also vary a value.

Example: Run PROC PRINT &PRTNUM times.

```
%MACRO PRTMAC(PRTNUM,NAME);
 %DO I= 1 %TO &PRTNUM;
     PROC PRINT DATA=&NAME&I;
      TITLE "PRINT OF DATASET &NAME&I";
RUN;
 %END;
%MEND;

%PRTMAC(4,PAYROLL)
```

The Generated SAS Code

```
PROC PRINT DATA=PAYROLL1;
  TITLE "PRINT OF DATASET PAYROLL1";
RUN;
PROC PRINT DATA=PAYROLL2;
  TITLE "PRINT OF DATASET PAYROLL2";
RUN;
PROC PRINT DATA=PAYROLL3;
  TITLE "PRINT OF DATASET PAYROLL3";
RUN;
PROC PRINT DATA=PAYROLL4;
  TITLE "PRINT OF DATASET PAYROLL4";
RUN;
```

Practice Exercises

Exercise 6:

- If the following macro was defined to the SAS system:

```
%MACRO FREQ(DSN,VAR1,VAR2);
  PROC FREQ DATA=&DSN;
    TABLES &VAR1*&VAR2 / NOPERCENT;
  RUN;
%MEND FREQ;
```

- What code would the SAS compiler see after this macro call?

```
%FREQ(FREQ,SALARIES,SALES)
```

Exercise 6 Solution

```
PROC FREQ DATA=FREQ;
  TABLES SALARIES*SALES / NOPERCENT;
RUN;
```

SAS DATA Step Interfaces

SYMGET, SYMPUT, and macro variables can transfer values between SAS steps.

Example: Display the number of observations in a dataset in a title.

```
%MACRO OBSCOUNT(NAME);
  DATA _NULL_;
   SET &NAME NOBS=OBSOUT;
   CALL SYMPUT('MOBSOUT',OBSOUT);
   STOP;
  RUN;
  PROC PRINT DATA=&NAME;
   TITLE "DATASET &NAME CONTAINS &MOBSOUT OBSERVATIONS";
  RUN;
%MEND;

%OBSCOUNT(PAYROLL)
```

The Generated SAS Code

```
DATA _NULL_;
 SET PAYROLL NOBS=OBSOUT;
 CALL SYMPUT('MOBSOUT',OBSOUT);
 STOP;
RUN;
PROC PRINT DATA=PAYROLL;
 TITLE  "DATASET PAYROLL CONTAINS 50 OBSERVATIONS";
RUN;
```

Notes:

- SYMGET returns macro variable values to the DATA step
- Macro variables created with SYMPUT, can be referenced via & in the ***NEXT*** step.

A SAS Macro Application

The following problem needs some help.

Data Set COUNTYDT

Obs	COUNTYNM	READING
1	ASHLAND	125
2	ASHLAND	611
3	BAYFIELD	101
4	BAYFIELD	101
5	BAYFIELD	222
6	WASHINGTON	143

Macro Application (continued)

Problem: Each day you read a SAS dataset containing data from counties in Wisconsin. Anywhere between 1 and 72 counties might report that day. Do the following:

1. Create a separate dataset for each reporting county.
2. Produce a separate PROC PRINT for each reporting county.
3. In the TITLE print the county name.
4. Reset the page number to 1 at the beginning of each report.
5. In a footnote print the number of observations processed for each county.

Question: How do you do it?

Solution: A Data Step and a SAS Macro.

A data step and a macro to generate the PROC PRINTs.

The data step goes through the data and:

- counts counties
- counts observations per county, puts in macro variables
- puts countynms into macro variables
- puts total counties reporting into a macro variable.

The Data Step Code

```
DATA _NULL_;
SET COUNTYDT END=EOF;              /* READ SAS DATASET      */
 BY COUNTYNM;                      /* SORT SEQ              */
 IF FIRST.COUNTYNM THEN DO;        /* NEW COUNTY ?          */
    NUMCTY+1;                      /* ADD 1 TO NUMCTY       */
    CTYOBS=0;                      /* OBS PER COUNTY TO 0   */
    END;
 CTYOBS+1;                         /* ADD ONE OBSER FOR CTY */
IF LAST.COUNTYNM THEN DO;          /* EOF CTY, MAKE MAC VARS*/
    CALL SYMPUT('MCTY'||LEFT(PUT(NUMCTY,3.)),COUNTYNM);
    CALL SYMPUT('MOBS'||LEFT(PUT(NUMCTY,3.)),LEFT(CTYOBS));
    END;
IF EOF THEN
    CALL SYMPUT('MTOTCT',NUMCTY);/* MAC VAR NO DIF CTYS  */
RUN;
%PUT *** MTOTCT=&MTOTCT;          /* DISPLAY NO OF CTYS   */
```

The Generated Macro Variables

One for each countynm, obs/county, and total num of counties.

```
      SYMBOL TABLE

NAME                VALUE

MCTY1               ASHLAND
MOBS1               2
MCTY2               BAYFIELD
MOBS2               3
MCTY3               WASHINGTON
MBOS3               1
MTOTCT              3
```

The Macro to Loop Around PROC PRINT

```
%MACRO COUNTYMC;                    /* MACRO START            */
%DO I=1 %TO &MTOTCT;                /* LOOP THRU ALL CTYS     */
 %PUT *** LOOP &I OF &MTOTCT;       /* DISPLAY PROGRESS       */
 PROC PRINT DATA=COUNTYDT;          /* PROC PRINT             */
  WHERE COUNTYNM="&&MCTY&I";        /* GENERATED WHERE        */
  OPTIONS PAGENO=1;                 /* RESET PAGENO           */
  TITLE "REPORT FOR COUNTY &&MCTY&I";
                                        /* TITLES AND FOOTNOTES  */
  FOOTNOTE "TOTAL OBSERVATION COUNT WAS &&MOBS&I";
 RUN;
%END;                               /* END OF %DO             */
%MEND COUNTYMC;                     /* END OF MACRO           */
%COUNTYMC                           /* INVOKE MACRO           */
```

The Generated Code

```
*** MTOTCT=3
*** LOOP 1 OF 3
    PROC PRINT DATA=COUNTYDT;
    WHERE COUNTYNM="ASHLAND";   OPTIONS PAGENO=1;
    TITLE "REPORT FOR COUNTY ASHLAND";
    FOOTNOTE "TOTAL OBSERVATION COUNT WAS 2";   RUN;
*** LOOP 2 OF 3
    PROC PRINT DATA=COUNTYDT;
    WHERE COUNTYNM="BAYFIELD";   OPTIONS PAGENO=1;
    TITLE "REPORT FOR COUNTY BAYFIELD";
    FOOTNOTE "TOTAL OBSERVATION COUNT WAS 3";   RUN;
*** LOOP 3 OF 3
    PROC PRINT DATA=COUNTYDT;
    WHERE COUNTYNM="WASHINGTON";   OPTIONS PAGENO=1;
    TITLE "REPORT FOR COUNTY WASHINGTON";
    FOOTNOTE "TOTAL OBSERVATION COUNT WAS 1";  RUN;
```

The Generated Output

```
REPORT FOR COUNTY ASHLAND        1

OBS     COUNTYNM     READING

 1      ASHLAND        125
 2      ASHLAND        611

TOTAL OBSERVATION COUNT WAS 2
```

```
REPORT FOR COUNTY BAYFIELD       1

OBS     COUNTYNM     READING

 3      BAYFIELD       101
 4      BAYFIELD       101
 5      BAYFIELD       222

TOTAL OBSERVATION COUNT WAS 3
```

The Generated Output

```
REPORT FOR COUNTY WASHINGTON     1

OBS      COUNTYNM      READING

  6      WASHINGTON       143

TOTAL OBSERVATION COUNT WAS 1
```

A Solution Via PROC SQL

PROC SQL can create macro variables.

```
PROC SQL;
  SELECT  LEFT(PUT(COUNT(DISTINCT COUNTYNM),3.))
           INTO:MTOTCT FROM COUNTYDT;
                                  /* NO OF UNIQUE CTYS     */
  SELECT  DISTINCT COUNTYNM       /* EACH CTY NAME         */
          INTO:MCTY1-:MCTY&MTOTCT FROM COUNTYDT;
  SELECT  COUNT(*)                /* OBS PER               */
          INTO:MOBS1-:MOBS&MTOTCT FROM COUNTYDT
  GROUP BY COUNTYNM;

%PUT *** MTOTCT=&MTOTCT;          /* DISPLAY NO OF CTYS    */
```

The Macro Is Unchanged

```
%MACRO COUNTYMC;                    /* MACRO START            */
%DO I=1 %TO &MTOTCT;                /* LOOP THRU ALL CTYS     */
 %PUT *** LOOP &I OF &MTOTCT;       /* DISPLAY PROGRESS       */
 PROC PRINT DATA=COUNTYDT;          /* PROC PRINT             */
  WHERE COUNTYNM="&&MCTY&I";        /* GENERATED WHERE        */
  OPTIONS PAGENO=1;                 /* RESET PAGENO           */
  TITLE "REPORT FOR COUNTY &&MCTY&I";
                                    /* TITLES AND FOOTNOTES   */
 FOOTNOTE "TOTAL OBSERVATION COUNT WAS &&MOBS&I";
 RUN;
%END;                               /* END OF %DO             */
%MEND COUNTYMC;                     /* END OF MACRO           */
%COUNTYMC                           /* INVOKE MACRO           */
```

Notes:

- PROC SQL must produce a report when creating macro variables.
- PROC PRINTTO could route it to a null file.

Practice Exercises

Exercise 7:

- A very common problem is to run a proc against all members of a SAS library.
- The following program produces a SAS dataset containing an observation for each variable within each dataset in the library. A partial print is shown below.

```
 PROC CONTENTS DATA=WORK._ALL_ NOPRINT OUT=CONTOUT;
  RUN;
 PROC PRINT DATA=CONTOUT;
 VAR LIBNAME MEMNAME NAME;
 TITLE 'CONTOUT';
RUN;
```

The Resulting Dataset

CONTOUT

OBS	LIBNAME	MEMNAME	NAME
1	WORK	FREQ	DEPT
2	WORK	FREQ	SALARIES
3	WORK	FREQ	SALES
4	WORK	FREQ	YEAR
5	WORK	YEAR1988	EMPLOYDT
6	WORK	YEAR1988	EXPENSES
7	WORK	YEAR1988	NAME
8	WORK	YEAR1988	REGION
9	WORK	YEAR1988	SALES
10	WORK	YEAR1988	STATE

Directions:

- Write a SAS macro that will generate a separate PROC PRINT of all members in the library with an appropriate title.
- Note you will only want to produce one print per member, not one per variable.

Exercise 7 Solution (first step)

```
PROC CONTENTS DATA=WORK._ALL_ NOPRINT OUT=CONTOUT;
 RUN;
 PROC PRINT DATA=CONTOUT;
 VAR LIBNAME MEMNAME NAME;
 TITLE 'CONTOUT';
RUN;
```

Exercise 7 Solution (remainder)

```
DATA ONEMEM;
 SET CONTOUT END=EOF;
  BY MEMNAME;
  IF LAST.MEMNAME;
  KTR+1;
  CALL SYMPUT ('MMEM'||LEFT(PUT(KTR,5.)),MEMNAME);
  IF EOF;
  CALL SYMPUT ('MTOTOBS',LEFT(PUT(KTR,5.)));
RUN;
%MACRO PRTLOOP;
 %DO I = 1 %TO &MTOTOBS;
     PROC PRINT DATA=&&MMEM&I;
        TITLE "&&MMEM&I";
     RUN;
 %END;
%MEND PRTLOOP;
%PRTLOOP
```

The Generated SAS Code

```
PROC PRINT DATA=FREQ ;
 TITLE "FREQ    ";
 RUN;
PROC PRINT DATA=YEAR1998;
 TITLE "YEAR1998";
 RUN;4
```

The Generated Output

FREQ OBS	DEPT	YEAR	SALARIES	SALES
1	1	97	1000	1000
2	1	98	1000	2000
3	2	97	500	3000
4	2	98	600	2000

The Generated Output

YEAR1998

OBS	NAME	STATE	REGION	EMPLOYDT	SALES	EXPENSES
1	ANDERSON, CHRIS	FL	1	02/08/93	56,000	12,000
2	PHILLIPS, HENRY	TX	1	04/14/96	63,432	23,500
3	ANDERSEN, JANET	WI	4	06/15/95	101,000	25,000
4	FRANK, TIMOTHY	FL	3	03/02/96	95,900	10,000
5	WILSON, MARGARET	TX	2	06/01/98	15,000	5,000
6	JONES, JACKIE	CA	1	03/14/96	35,000	12,000
7	SMITH, WILLIAM	WI	3	07/01/97	66,666	6,666
8	WILLIAMS, STEVEN	TX	2	04/01/95	43,000	10,000
9	JOHNSON, JOY	CA	1	05/10/96	20,000	4,000
10	MATHERS, MARK	WI	3	01/15/94	55,555	13,500
11	JENSEN, LORI	FL	3	10/10/96	103,500	30,000
12	O'HARE, PATTY	WI	1	02/28/98	25,000	500
13	FRANKLIN, SUSAN	CA	1	09/30/92	95,000	15,000
14	HARRISON, ANTHONY	WI	2	11/15/95	45,900	10,000
15	THOMPSON, ELIZABETH	FL	3	12/10/97	10,000	500

The SSCFLAT Macro

A general purpose macro:

- converts any SAS ds to a comma delimited file for input to spreadsheets etc.
- will run on all platforms
- automatically reads dictionary tables to get ds definition
- honors SAS formatting
- can create a header line
- dropping, reordering variables etc. can be done in an earlier SAS step.

The SSCFLAT Macro Partial Source

```
/*************************************************************/
/* MACRO SSCFLAT VERSION 1.4                                 */
/* CREATES A COMMA DELIMITED FILE FROM ANY SAS DATA SET      */
/* IT CAN BE THEN DOWNLOADED & IMPORTED INTO A SPREADSHEET   */
/*                                                           */
/* SAMPLE WINDOWS CALL:                                      */
/*  %SSCFLAT(MSASDS=MAIL.TEMPMAIL.MPREFIX=C:\TEMP\)          */
/*                                                           */
/* SAMPLE MVS CALL:                                          */
/*  %SSCFLAT(MSASDS=WORK.XYZ.MPREFIX=MYUSER)                 */
/*                                                           */
/* STEVEN FIRST                                              */
/* (C) SYSTEMS SEMINAR CONSULTANTS 1999   608 278-9964       */
/*                                                           */
/* PERMISSION GIVEN TO FORMER SSC SAS STUDENTS TO USE        */
/*     IN PERSONAL SAS JOBS.                                 */
/* FOR PERMISSION TO DISTRIBUTE TO OTHERS, OR FOR            */
/*     COMPANY WIDE USE, CONTACT SSC AT 608 278-9964         */
/*                                                           */
/*************************************************************/
```

SSCFLAT Macro Partial Source (continued)

```
%MACRO SSCFLAT(MSASDS=,                    /*INPUT SAS DS (REQUIRED        */
   MPREFIX=&SYSPREF..,                     /*OUT PREF, OR DIR OUT          */
   MFLATOUT=&MPREFIX&MMEMNAME..DAT,        /*FLATFILE OUT                  */
   MHEADER=YES,                            /*FIELD NAMES IN FIRST REC      */
   MLIST=YES,                              /*PRINT FLAT FILE IN LOG?       */
   MTRIMNUM=YES,                           /*TRIM NUM TRAIL BLANKS?        */
   MTRACE=NO,                              /*DEBUGGING OPTION              */
   MMISSING=".",                           /*MISSING VALUE CHARACTER       */
   MLRECL=6160,                            /*LARGEST RECORD LENGTH         */
   MVSOPT=UNIT=3390,                       /*MVS UNIT OPTIONS              */
   MSPACE=1,                               /*MVS SPACE IN CYLS             */
   );

%PUT ***** SSCMAC COPYRIGHT (C) 1999 SYSTEMS SEMINAR;
%PUT ***** CONSULTANTS 608 278-9964;
```

A SSCFLAT Macro Example

In windows, convert WORK.ADDRDATA to a FLAT file.

ADDRDATA

OBS	NAMES	DEPT	AGE	RATE
1	STEVE	ACCT	43	12.22
2	DAVID	PAYR		11.21

```
%INC 'insert file ref\sscflat.sas.';
%SSCFLAT(MSASDS=WORK.ADDRDATA,
    mprefix=c:\temp\)   *invoke macro;
```

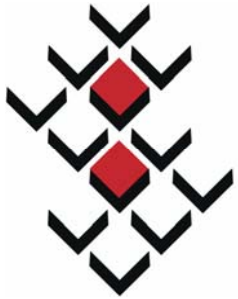
A SSCFLAT Macro Example (continued)

The c:\temp\addrdata.dat flat file created:

```
“NAME”,“DEPT”,“AGE”,“RATE”
“STEVE”,”ACCT ”,43,12.22
“DAVID”,”PAYR “,,11.21
```

The flat file is now available for download and/or import to spreadsheets, etc.

Contact Us



SYSTEMS SEMINAR CONSULTANTS, INC.

SAS® Training, Consulting, & Help Desk Services

Steven J. First

President

(608) 278-9964, Ext. 302

FAX (608) 278-0065

sfirst@sys-seminar.com

www.sys-seminar.com

2997 Yarmouth Greenway Drive • Madison, WI 53711

QUALITY PARTNER™ SAS INSTITUTE INC.