Abstract

The SAS programming language has a rich "tool-box" of features that can offer a lot of power to the user. The ability to use macro variables and macro code is difficult to learn without some training and examples that are easy to understand. This tutorial will introduce the user to a few of the very powerful techniques possible using macro variables and macro programming.

Objectives:

After this tutorial attendees 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.

Traditional SAS Programming

Without macros, SAS programs are a just series of DATA and PROC steps.

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

Step boundaries are the SAS keywords:

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

Notes:

Each step is compiled and executed independently.
All SAS jobs have a step boundary at the beginning.
Batch and non-interactive jobs have an implied ENDSAS at end of the SYSIN file.
RUN and QUIT have special considerations.

Step boundaries control the compilation and execution of SAS.

```
data saleexps;          <- Step, start compile
  infile rawin;
  input name $1-10 division $12
       years 15-16 sales 19-25
       expense 27-34;  <- Step, exec , start compile
  proc print data=saleexps;
  proc means data=saleexps;
  var sales expense;    <- EOF (batch), exec, ENDSAS.
```

Notes:

Interactive jobs have no end-of-file, so the
The RUN Statement
RUN acts as an explicit step boundary in most PROCs.

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

Notes:
The use of RUN after each step is highly recommended.

SAS Global Statements
Global statements are executed immediately.

filename rawin 'saleexps.dat';  < comp, exec
  data saleexps;
  input name $1-10 division $12
  years 15-16 sales 19-25
  expense 27-34;
run;                           < end, exec
proc print data=saleexps;
  options ls=80 nodate;
  title 'sample title';
run;
proc means data=saleexps;
  var sales expense;
run;

Notes:
RUN ensures global statements affect the correct step.

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

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.
Beginning Tutorials

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 functions:
- help process and evaluate text and macro variables
- have some capabilities of SAS data step functions
- have some unique capabilities.

Macro Language Components (continued)

Macro expressions:
- are sequences of text linked with operators and parentheses
- are needed by some macro functions and programming statements.

Macro constants:
- are treated as character strings
- can contain literals, variable names, numbers, dataset names, SAS statements.

A Macro 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.

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

<table>
<thead>
<tr>
<th>Department Sales as of Wednesday 04JAN89</th>
<th>Deliver to Michael O'Malley</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBS</td>
<td>DEPT</td>
</tr>
<tr>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Notes:
- Macro variables are NOT resolved within double quotes.
- Macro variables can be used in almost any SAS statement.

Partial List of Automatic Macro Variables
- SYSBUFFR text entered in response to %INPUT
- SYSCMD last non-SAS command entered
- SYSDATE current date DATE6., DATE7. format
- SYSDAY current day of the week
- SYSDAYV current graphics device
- SYSDSN last da Ex. WORK SOFTSALE
- SYSENV SAS environment (FORE or BACK)
- SYSERR return code set by SAS procedures
- SYSFILRC FILENAME rc
- 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 Ex. WORK SOFTSALE
- SYSLIBRC rc from last LIBNAME statement
- SYSLCKRC whether most recent lock was successful
- SYSMENV macro environment
- SYSMSG message displayed with %DISPLAY
- SYSWRP 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 "REPORT WAS RUN ON $$\&$$SYSDAY, $$\&$$SYSDATE";

Resolves to:

FOOTNOTE "REPORT WAS RUN ON MONDAY, 17FEB97";

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

Syntax:

%PUT text macrovariables _all_

Example:

DATA NEWPAY;
INPUT EMP$ RATE;
DATALINES;
TOM 10
JIM 10
; RUN;
datasets were built correctly.

Exercise 2:

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

&SYSNAME: Current day of the week
&SYSNAME: 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.

A Macro 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;

You would like to:

☐ be able to change the name quickly in one place only
☐ have the datasetname appear in a title.

How can you do the above?

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;

Resolves to:

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;

Resolves to:

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

Assigning SAS Statements to Macro Variables
%STR allows values with ; etc.

%LET NAME=NEWPAY;
%LET CHART=STR(100;100;20000;20000;5;5;1;1;1;1;5;5;1;1;1;1;5;5;1;1;
  1;1;1;1;5;5;1;1;1;1;1;1;
  TOM 10
  JIM 10
  ;
  &CHART
PROC PRINT DATA=&NAME;
  TITLE "PRINT OF DATASET &NAME";
RUN;

Resolves to:

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;

Nesting of Macro Variables
Macro variables can contain other macro variables.

%LET NAME=NEWPAY;
%LET CHART=STR(100;100;20000;20000;5;5;1;1;1;1;5;5;1;1;1;1;5;5;1;1;
  1;1;1;1;5;5;1;1;1;1;1;1;
  DATA &NAME;
  INPUT EMP$ RATE;
  DATALINES;
  TOM 10
  JIM 10
  ;
  &CHART
PROC PRINT DATA=&NAME;
  TITLE "PRINT OF DATASET &NAME";
RUN;

Resolves to:

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
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=1989;
%LET C=CANES;
%LET D=DECEMBER 31;
%LET E="TREMENDOUS";

What would be the results of these %PUT statements? (Include all spaces).

%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!

Practice Exercises

Exercise 5:

After execution of the following %LET statements

%LET A=WIND;
%LET B=WATER;
%LET C=SURFING;
%LET &C=RUN;
%LET D=AND;
%LET E=MAN;
%LET &E=STR(SNOW);
%LET F=STR(PROC PRINT);
%LET G=STR(RUN);

What would be the results of these %PUT statements?
(Include all spaces).

%PUT &A&C;

%PUT &A &D &B &C = &SURFING;

%PUT H&D.Y &E;

%PUT H&D.Y &E;

%PUT H&D.Y.&E;

%PUT &C &&C;

%PUT &MAN&EE;

%PUT &F &G;

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;

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

Resolves to:
DATA NEWPAY;
INPUT EMPS RATE;
DATALINES;
TOM 10
JIM 10
; RUN;
PROC CHART DATA=NEWPAY;
VBAR EMP;
RUN;
PROC PRINT DATA=NEWPAY;
TITLE "PRINT OF DATASET NEWPAY";
RUN;

Positional Macro Parameters

Macro parameters are defined in order after the macro name.

%MACRO CHART(NAME,VARVAR);
PROC CHART DATA=&NAME;
VBAR &VARVAR;
RUN;
%MEND;
%CHART(PAYROLL,EMP)

Resolves to:
PROC CHART DATA=PAYROLL;
VBAR EMP;
RUN;

Notes:
Keyword parameters are also allowed. Keyword parameters can give default values.
Nested Macros
Macros can call other macros.

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

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

%PCHART(PAYROLL,EMP)

Resolves to:

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

Iterative Macro Invocation
%DO can also vary a value.

Example: Run PROC PRINT &PRTNUM times.

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

%PRTMAC(4,PAYROLL)

Resolves to:

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;
   PROC FREQ DATA=&DSN; 
   TABLES &VAR1*&VAR2 / NOPERCENT;
   RUN;
%MEND FREQ;

What code would the SAS compiler see after these statements?

%LET DSN=FREQ;
%LET VAR1=DEPT;
%LET VAR2=SALES;
%FREQ

Practice Exercises

Exercise 7:
If the following macro was defined to the SAS system
%MACRO FREQDSN,VAR1,VAR2;
PROC FREQ DATA=DSN; TABLES &VAR1*&VAR2 / NOPERCENT;
RUN;
%MEND FREQ;

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

A) %FREQ(FREQ,SALES,SALES)

SAS would stop processing the code generated by the following calls. Can you determine why?

B) %FREQ()

C) %FREQ(FREQ,SALES,)

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)

Resolves to:

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.

A SAS Macro Application
The following problem needs some help.

<table>
<thead>
<tr>
<th>Data Set COUNTYDT</th>
<th>Obs</th>
<th>COUNTYNM</th>
<th>READING</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ASHLAND</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ASHLAND</td>
<td>611</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BAYFIELD</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>BAYFIELD</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>BAYFIELD</td>
<td>222</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>WASHINGTON</td>
<td>143</td>
<td></td>
</tr>
</tbody>
</table>

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.

How do you do it?
Solution: Write a SAS Macro.

The Solution
A multi-step macro can communicate between steps.

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 /* VERY LAST OBS ? */
   CALL SYMPUT('MTOTCT',NUMCTY); /* MAC VAR NO DIF CTYS */
RUN;
/*PUT *** MTOTCT=&MTOTCT; */ /* DISPLAY NO OF CTYS */
/*MACRO COUNTYM: */ /* MACRO START */
/*DO 1=1 %TO &MTOTCT; */ /* LOOP THRU ALL CTYS */
/*PUT *** LOOP &1 OF &MTOTCT; */ /* DISPLAY PROGRESS */
/*PROC PRINT DATA=COUNTYDT; */ /* PROC PRINT */
/*WHERE COUNTYNM="&COUNTYM"; */ /* GENERATED WHERE */
/*OPTIONS PAGENO=1; */ /* RESET PAGENO */
/*TITLE "REPORT FOR COUNTY &COUNTYM"; */ /* TITLES AND FOOTNOTES */
/*FOOTNOTE "TOTAL OBSERVATION COUNT WAS &MOBS&1";*/
RUN;
%MEND;
/*END OF DO */
%MEND COUNTYM; /* END OF MACRO */
%COUNTYM /* INVOKE MACRO */
BEGINNING TUTORIALS

THE GENERATED CODE AND OUTPUT

*** MTOCT=3
*** LOOP 1 OF 3
PROC PRINT DATA=COUNTYDT;
WHERE COUNTYMN="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 COUNTYMN="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 COUNTYMN="WASHINGTON"; OPTIONS PAGENO=1;
TITLE "REPORT FOR COUNTY WASHINGTON"
FOOTNOTE "TOTAL OBSERVATION COUNT WAS 1"; RUN;

REPORT FOR COUNTY ASHLAND

OBS COUNTYMN READING
1 ASHLAND 125
2 ASHLAND 611
TOTAL OBSERVATION COUNT WAS 2

REPORT FOR COUNTY BAYFIELD

OBS COUNTYMN READING
3 BAYFIELD 101
4 BAYFIELD 101
5 BAYFIELD 222
TOTAL OBSERVATION COUNT WAS 3

REPORT FOR COUNTY WASHINGTON

OBS COUNTYMN READING
6 WASHINGTON 143
TOTAL OBSERVATION COUNT WAS 1

VAR LIBNAME MENNAME NAME;
TITLE 'CONTOUT';
RUN;

CONTOUT

OBS LIBNAME MENNAME 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

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 SOLUTIONS

Exercise 1

%inc 'a:\sasm0001.sas';
proc contents data=work._all_;run;

Exercise 2

(The following code can be entered to see the results of these System Macro variables)

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

Exercise 3

proc contents data=&syslast;
Title "Contents of &syslast";
run;

Exercise 4

A) CANES
B) FISCAL YEAR 1989
C) YEAR ENDED DECEMBER 31, 1989
D) 1989 CANDY CANES WERE SOLD IN 1989
E) 1989 WAS A "TREMENDOUS" SALES YEAR!

Exercise 5

A) WIND SURFING
B) WIND AND WATER SURFING = FUN
C) H8DY MAN WARNING:
Apparent symbolic reference DY not resolved.
D) H8DYN MAN
E) H8DYN MAN

328
F) SURFING FUN

G) SNOW MAN

H) PROC PRINT; RUN;

Exercise 6

PROC FREQ DATA=FREQ;
   TABLES DEPT*SALES / NOPERCENT; RUN;

Exercise 7

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

The following code would be passed to the SAS compiler
and would be rejected due to syntax error:

B) PROC FREQ DATA=;
   TABLES * / NOPERCENT;
   RUN;

C) PROC FREQ DATA=FREQ;
   TABLES SALARIES* / NOPERCENT;
   RUN;

Exercise 8

PROC CONTENTS DATA=WORK..ALL_ NOPRINT OUT=CONTOUT;
   RUN;
PROC PRINT DATA=CONTOUT;
VAR LIBNAME MEmNAME NAME;
TITLE 'CONTOUT';

RUN;
DATA OMEM;
SET CONTOUT END=EOF;
BY MEmNAME;
IF LAST.MEmNAME;
   KTR=1;
   CALL SYMPUT ('MEM'||LEFT(PUT(KTR,5.)),MEmNAME);
IF EOF;
   CALL SYMPUT ('MTOTOSBS',LEFT(PUT(KTR,5.)));
RUN;
%MACRO PRITLOOP;
%DO I = 1 %TO &MTOTOSBS;
   PROC PRINT DATA=&MEMM&E;
   TITLE "&MEMM&E";
   RUN;
%END;
%MEND PRITLOOP;
%PRITLOOP

Steven First
Systems Seminar Consultants
2997 Yarmouth Greenway
Madison, WI 53711
Phone: (608) 278-9964
Fax: (608) 278-0065
E-mail: sfirst@sys-seminar.com