

Debugging

Where to start?

John Ladds,
SAS Technology Center, Statistics Canada

Did it work?

“I don’t see any red.
So it must have worked, right?”



Clear the log and output

```
dm log 'clear' log;  
dm out 'clear' wpgm;
```

- Read the current process' LOG
- Clear old log and output windows
 - Don't try to solve a non-problem



The PUT statement

- PUT “Started Step 1”;
- PUT “Ended Step1”;
 - Did all the steps run that you expected
- PUTLOG x=;
 - Directs out to the log even if you are sending other output to the output window
- PUT ‘x=’ x; is the same as PUT x=;



%PUT Statements

- Generally for macro coding
 - But can be used anywhere
 - PUT can only be used in a data step
 - %PUT _ALL_; All macro variables
 - %PUT _USER_; User defined only
 - %PUT _LOCAL_; Local only
 - %PUT _GLOBAL_; Global only



Macro Options

- MPRINT / NOMPRINT
- MLOGIC / NOMLOGIC
- SYMBOLGEN / NO SYMBOLGEN



Simple Macro Code

```
%macro Hi(name=);
```

```
  %if "&name." = ""
```

```
    %then %do;
```

```
      data _null_;
```

```
        putlog 'Hello world';
```

```
      run;
```

```
    %end;
```

```
  %else %do;
```

```
    data _null_;
```

```
      putlog "Hello &name.";
```

```
    run;
```

```
  %end;
```

```
%mend Hi;
```



MPRINT Option

```
263 options mprint nomlogic nosymbolgen;
```

```
264 %Hi(name=)
```

```
MPRINT(HI): data _null_;
```

```
MPRINT(HI): putlog 'Hello world';
```

```
MPRINT(HI): run;
```

Hello world

```
265 %Hi(name=OASUS)
```

```
MPRINT(HI): data _null_;
```

```
MPRINT(HI): putlog "Hello OASUS";
```

```
MPRINT(HI): run;
```

Hello OASUS



MLOGIC Option

```
286 options nomprint mlogic nosymbolgen;  
287 %Hi(name=)
```

```
MLOGIC(HI): Beginning execution.  
MLOGIC(HI): Parameter NAME has value  
MLOGIC(HI): %IF condition "&name." = "" is TRUE
```

```
Hello world  
MLOGIC(HI): Ending execution.  
288 %Hi(name=OASUS)
```

```
MLOGIC(HI): Beginning execution.  
MLOGIC(HI): Parameter NAME has value OASUS  
MLOGIC(HI): %IF condition "&name." = "" is FALSE
```

```
Hello OASUS  
MLOGIC(HI): Ending execution.
```



SYMBOLGEN Option

309 options nomprint nomlogic symbolgen;

310 %Hi(name=)

SYMBOLGEN: Macro variable NAME resolves to

Hello world

SYMBOLGEN: Macro variable NAME resolves to OASUS

311 %Hi(name=OASUS)

SYMBOLGEN: Macro variable NAME resolves to OASUS

Hello OASUS



Improving your code

- Include comments that explain what you are trying to accomplish
- Neatness counts
 - Line up lists
 - Follow a set style
 - i.e. IF-THAN-ELSE indenting
 - Add a comment to the END statement of a long DO block.
 - END; /* of month processing */



Improving your code

- `%*` vs `*%`
 - `%*` comment; is a macro comment
 - `*%` comment; is a base SAS comment
 - but they look the same in your code.
 - If you use this style to skip macro code the statement may still run anyway.



Build a model

- Create a model of a specific problem
 - Separate the problem from the rest of the code
 - Reduce the number of observations if the data is too large.
 - IF `_N_ LE 500;`
 - select the first 500 observations
 - IF `MOD(_n_, 500)=0;`
 - select every 500th observation



SAS/Access Problems

```
options sastrace=',,,d'
```

```
sastraceloc=saslog nostsuffix;
```

- These options are found in the SAS/Access section of the online help.



SAS/Access Problems

NOTE: Libref TSTDATA was successfully assigned as follows:

Engine: ODBC

Physical Name: cdbwr

NOTE: Table WORK.TEST_DATA_FILE created, with 16368 rows and 9 columns.

NOTE: PROCEDURE SQL used (Total process time):

real time 0.34 seconds

cpu time 0.06 seconds



SAS/Access Problems

NOTE: Libref TSTDATA was successfully assigned as follows:

Engine: ODBC
Physical Name: cdbwr

ODBC: AUTOCOMMIT turned ON for connection id 0

ODBC_1: Prepared: on connection 0
SELECT * FROM tomkari.MON1001

ODBC_2: Prepared: on connection 0
SELECT * FROM tomkari.MON111

ODBC_3: Prepared: on connection 0
select t1."T", t1."S0387", t1."S0388", t1."S0389", t2."S391", t2."S409", t2."S418", t2."S427",
t2."S436" from tomkari.MON1001 t1, tomkari.MON111 t2

ODBC_4: Executed: on connection 0
Prepared statement ODBC_3

ACCESS ENGINE: SQL statement was passed to the DBMS for fetching data.

NOTE: Table WORK.TEST_DATA_FILE created, with 16368 rows and 9 columns.

NOTE: PROCEDURE SQL used (Total process time):

real time 0.31 seconds
cpu time 0.06 seconds

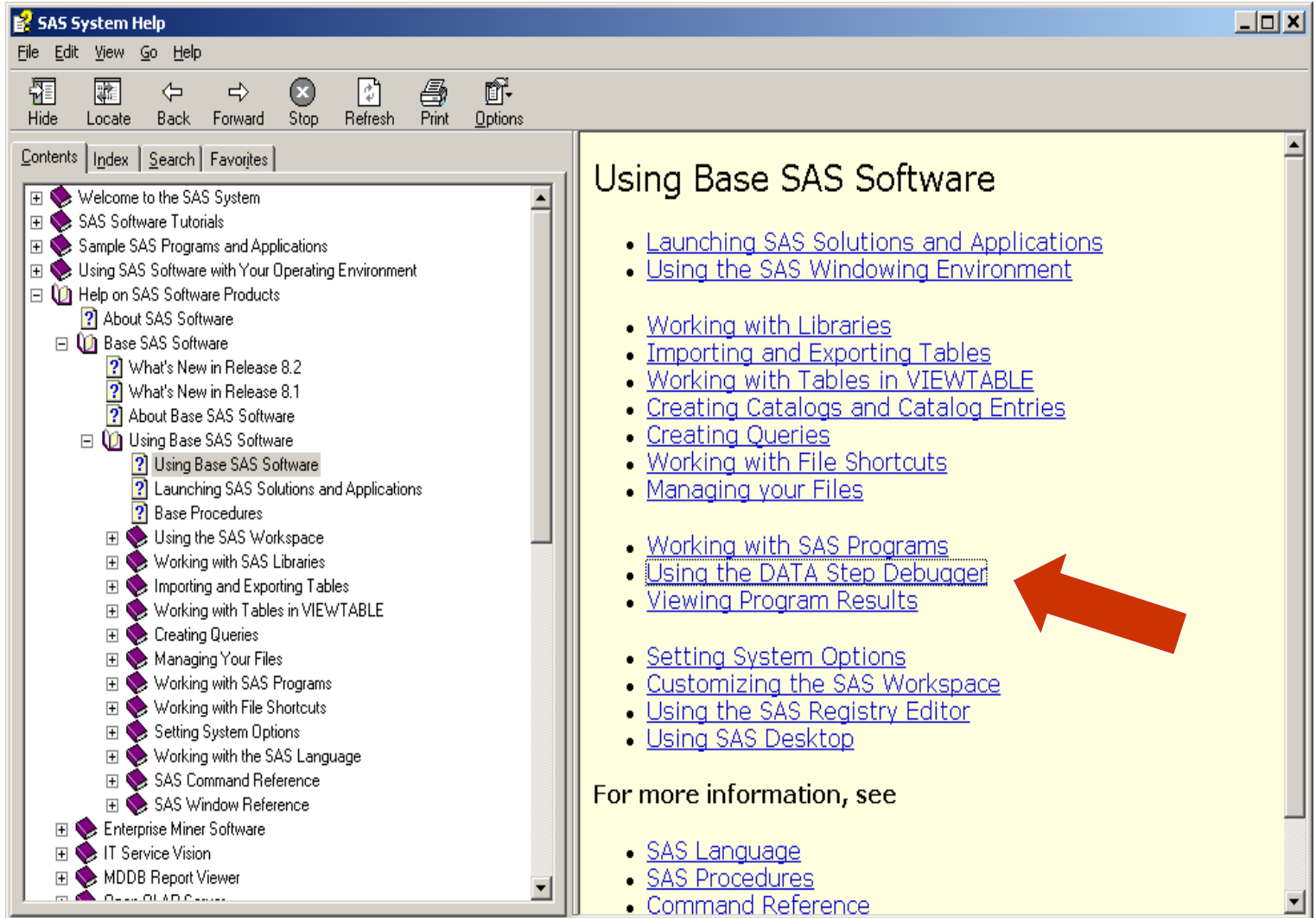


SAS Data Step Debugger

- Don't use / rewrite your code instead
- Not available in Enterprise Guide



SAS Data Step Debugger



The screenshot shows the SAS System Help window. The left pane displays a tree view of the help content, with 'Using Base SAS Software' expanded. The right pane shows the 'Using Base SAS Software' section, which contains a list of links. A red arrow points to the link 'Using the DATA Step Debugger'.

Using Base SAS Software

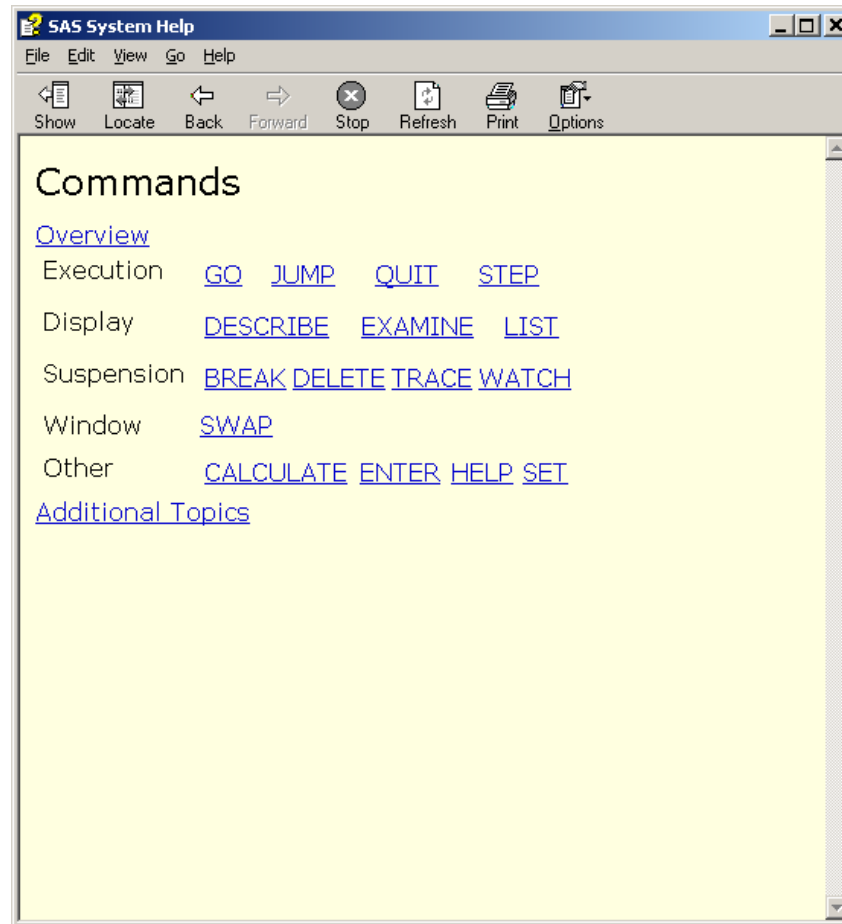
- [Launching SAS Solutions and Applications](#)
- [Using the SAS Windowing Environment](#)
- [Working with Libraries](#)
- [Importing and Exporting Tables](#)
- [Working with Tables in VIEWTABLE](#)
- [Creating Catalogs and Catalog Entries](#)
- [Creating Queries](#)
- [Working with File Shortcuts](#)
- [Managing your Files](#)
- [Working with SAS Programs](#)
- [Using the DATA Step Debugger](#)
- [Viewing Program Results](#)
- [Setting System Options](#)
- [Customizing the SAS Workspace](#)
- [Using the SAS Registry Editor](#)
- [Using SAS Desktop](#)

For more information, see

- [SAS Language](#)
- [SAS Procedures](#)
- [Command Reference](#)



SAS Data Step Debugger



Conclusion

- Read the log
 - Clear the log and output windows
- Use the LOG window to document steps
- Use options statements
 - Macro & SAS/Access
- Above all

“Neatness counts”

