

# An Animated guide to The Data Step Debugger

# Russell Lavery

# MACROS

# The Map of the SAS Macro System

## SAS Macro Quoting

# BASE SAS

## Proc Report

## Proc Summary

# SAS Arrays

## Proc Transpose

# The Data Step Debugger

## LARGE FILE TECHNIQUES: Speed Merging/Table Lookup

## SAS Resource Usage

## Key Merging and \_IORC\_ Formats

## Key-indexing, Bitmaps & Hashing

### Version 9 Hashing

# STATISITICS

## Matching of Test & Controls

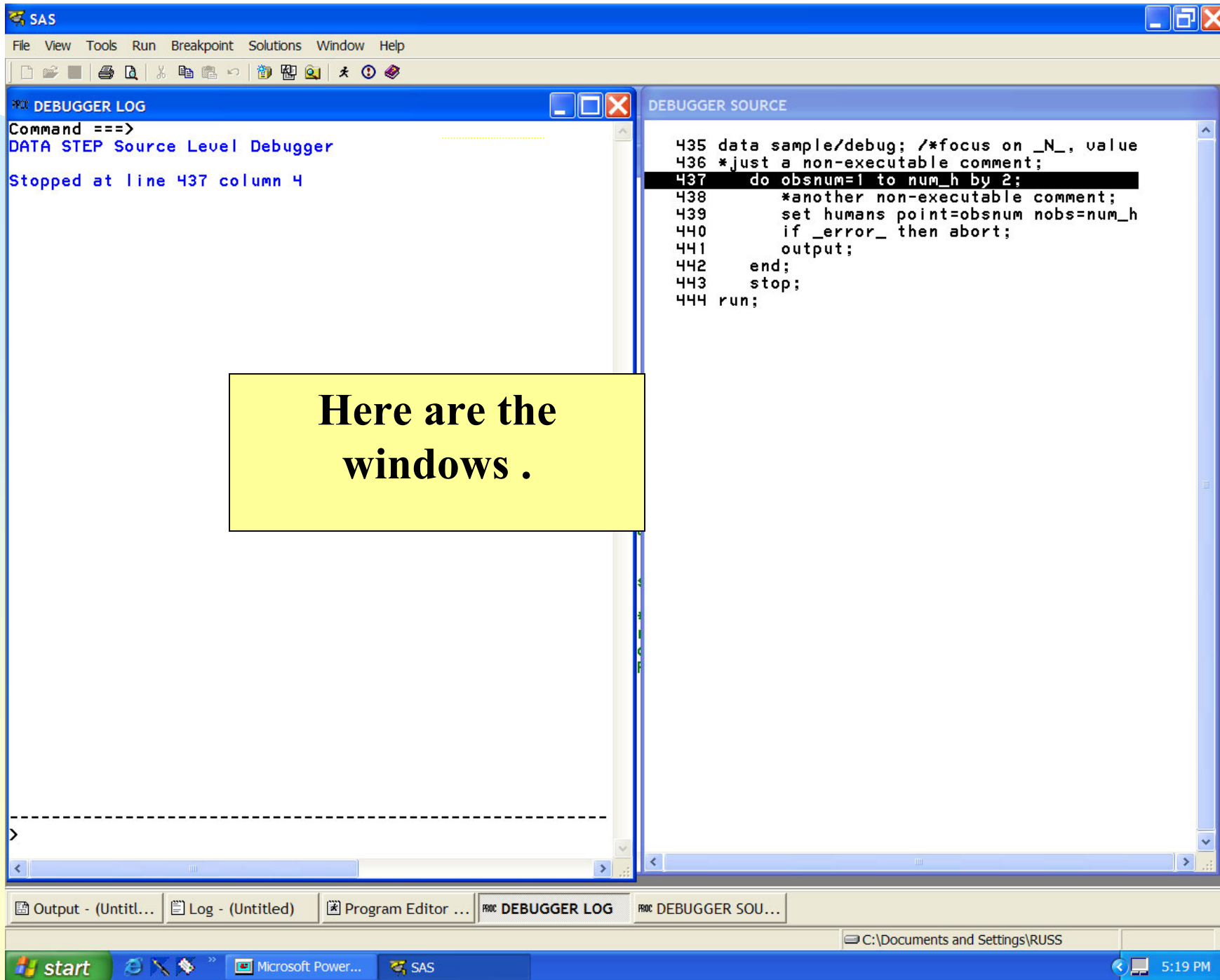
# The Logic of One Way Anova

# The logic of Regression

# The Logic of Hypothesis Testing

## Interactions and Two Way Anova

## Proc Power



# 5. Commands

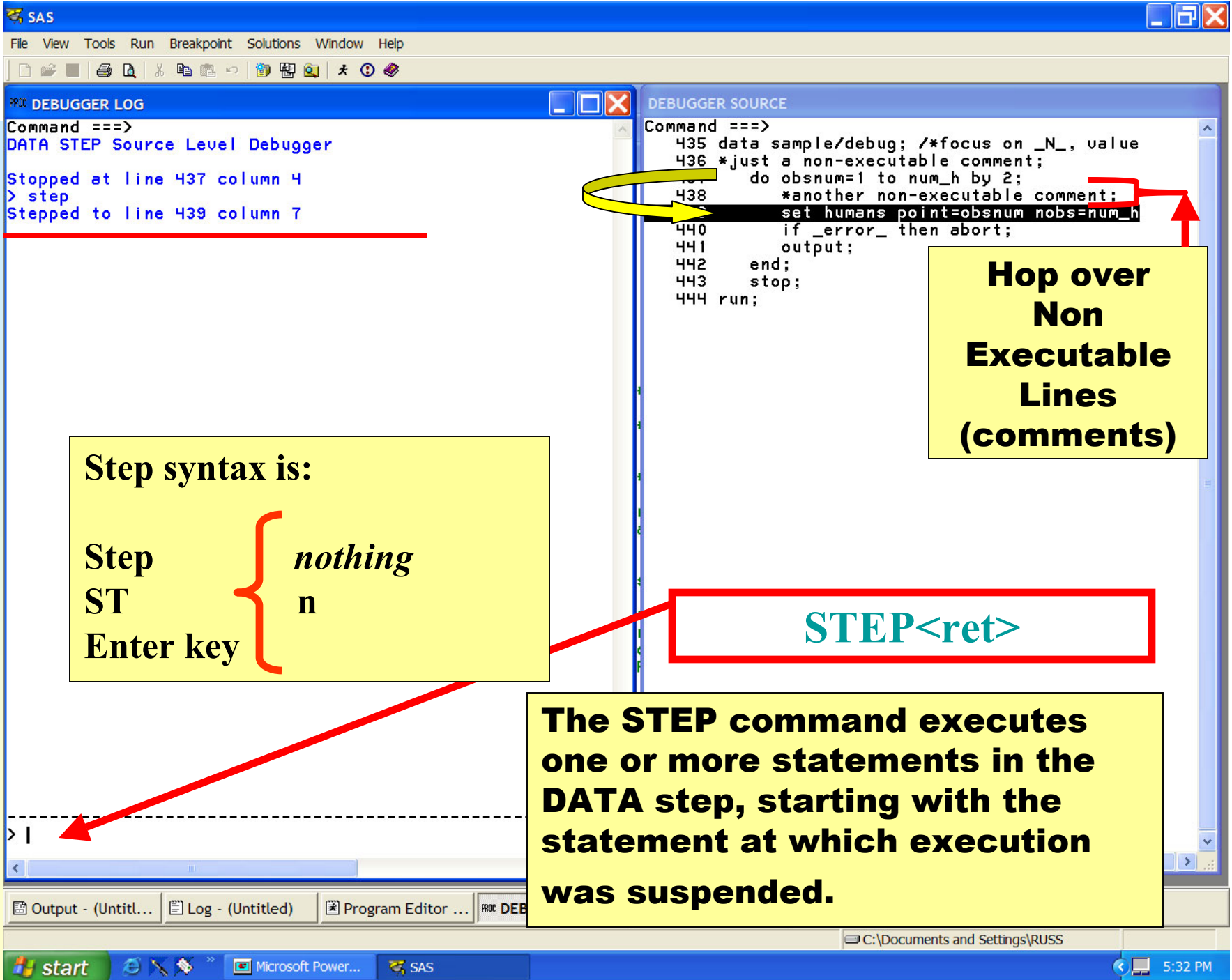
**Start by adding /Debug to data statement:  
Data First /Debug;**

**FIVE classes of commands**

<b>Execution</b>	<b>Step</b>	<b>Go</b>	<b>Quit</b>	<b>Jump</b>
<b>Display</b>	<b>Examine</b>	<b>List</b>	<b>Describe</b>	
<b>Suspension</b>	<b>Break</b>	<b>Delete</b>	<b>Watch</b>	<b>Trace</b>
<b>Window</b>	<b>Swap</b>			
<b>Other</b>	<b>Calculate</b>	<b>Set</b>	<b>Help</b>	

**SPEED TRICKS   Enter <F4>   Keys   Mouse   Macro**  
( commands available from SAS help facility)

---



The screenshot shows the SAS Debugger interface. The **DEBUGGER LOG** window on the left displays the following text:

```
Command ==>
DATA STEP Source Level Debugger

Stopped at line 437 column 4
> step
Stepped to line 439 column 7
> st 1
Stepped to line 440 column 7
>
Stepped to line 441 column 7
```

The **DEBUGGER SOURCE** window on the right shows the source code with line 441 highlighted:

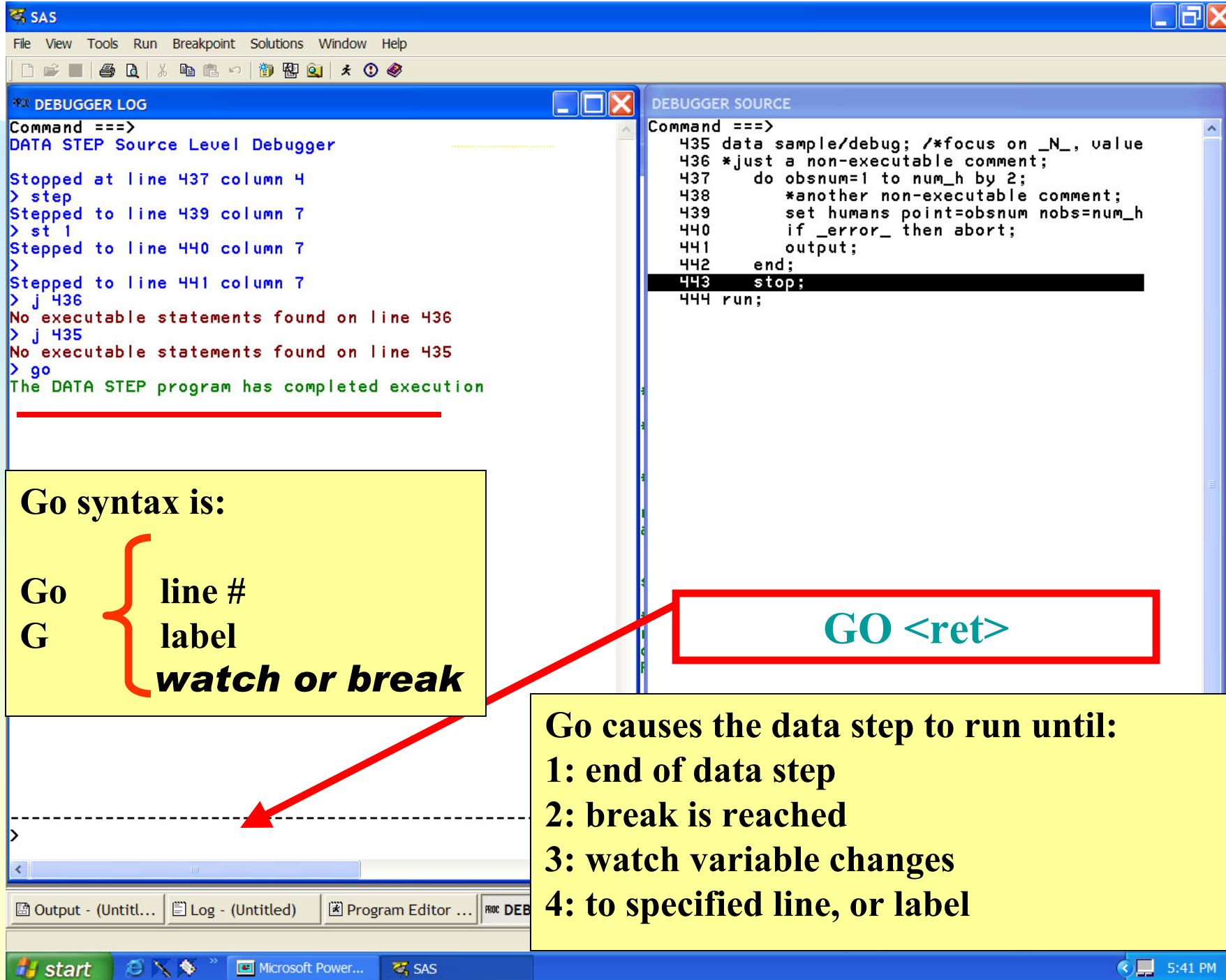
```
Command ==>
435 data sample/debug; /*focus on _N_, value
436 *just a non-executable comment;
437   do obsnum=1 to num_h by 2;
438     *another non-executable comment;
439     set humans point=obsnum nobs=num_h
440     if _error_ then abort;
441     output;
442   end;
443   stop;
444 run;
```

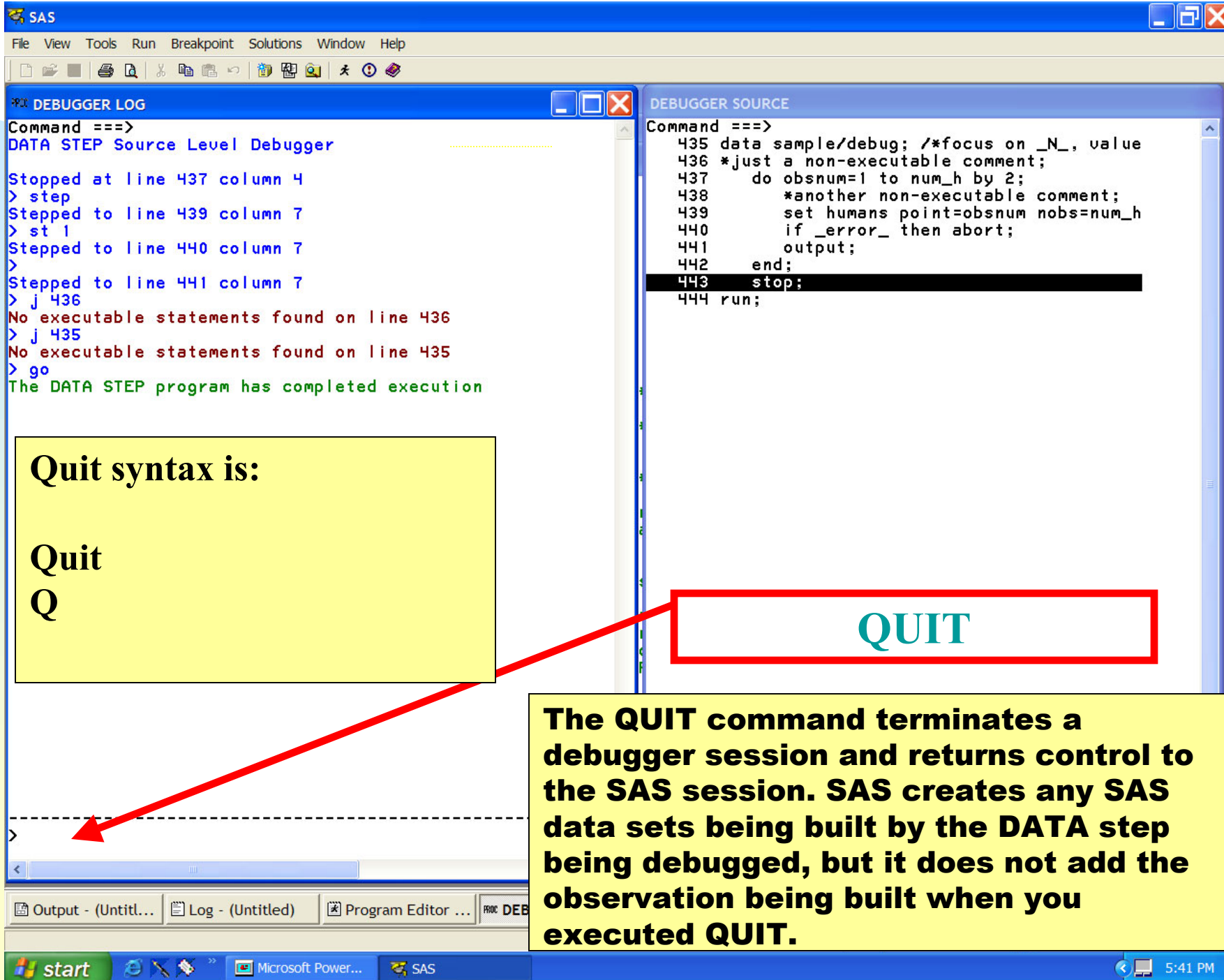
**Jump syntax is:**

**Jump**      { line #  
**J**            { label

**Do not use the JUMP command to jump to a statement inside a DO loop or to a label that is the target of a LINK-RETURN group. In such cases you bypass the controls set up at the beginning of the loop or in the LINK statement, and unexpected results may occur.**

**The JUMP command moves program execution to the specified location without executing intervening statements. After executing JUMP, you must restart execution with GO or STEP. You can jump to any executable statement in the DATA step. BUT..**







SAS

File View Tools Run Breakpoint Solutions Window Help

DEBUGGER LOG

Command ==>  
DATA STEP Source Level Debugger

Stopped at line 465 column 1  
> examine \_all\_  
name =  
repeat = .  
type =  
Pet\_nm =  
someday =  
\_ERROR\_ = 0  
\_N\_ = 1

**Values in PDV.  
Note all missing.  
Infile is about to execute.**

DEBUGGER SOURCE

```
464 data Prblm_pets (sortedby=name)/debug;  
465 infile datalines missover firstobs=2;  
466 *retain repeat;  
467 input @2 name $char5. @9 repeat 4.2 @15  
468                               @9 someday ;  
469 format someday date9. ;  
470 datalines;
```

**Examine syntax is:**

**examine** { **\_all\_**  
**E** **variable format**  
**var1 var2 var6**

**Note that the PDV is empty  
before the input statement  
executes!**

**Examine \_all\_**

**The EXAMINE command displays the  
value of one or more specified variables.  
The debugger displays the value using  
the format currently associated with the  
variable, unless you specify a different  
format.**

Output - (Untitl... Log - (Untitled) Program Editor ... PROC DEB

start Microsoft Power... SAS 5:47 PM



**SAS**

File View Tools Run Breakpoint Solutions Window Help

**DEBUGGER LOG**

```
Command ==>
DATA STEP Source Level Debugger

Stopped at line 465 column 1
> examine _all_
name =
repeat = .
type =
Pet_nm =
someday = .
_ERROR_ = 0
_N_ = 1
> st; ex _all_
Stepped to line 467 column 1
name =
repeat = .
type =
Pet_nm =
someday = .
_ERROR_ = 0
_N_ = 1
```

**DEBUGGER SOURCE**

```
Command ==>
464 data Prblm_pets (sortedby=name)/debug;
465 infile datalines missover firstobs=2;
466 *retain repeat;
467 input @2 name $char5. @9 repeat 4.2 @15
468 @9 someday ;
469 format someday date9.;
470 datalines;
```

**Examine syntax is:**

**examine** { **\_all\_** variable format var1 var2 var6 }

**Values in PDV. Note all missing. Input is about to execute.**

**Note that the PDV is empty before the input statement executes!**

**St; ex \_all\_**

**The EXAMINE command displays the value of one or more specified variables. The debugger displays the value using the format currently associated with the variable, unless you specify a different format.**

Output - (Untitl...) Log - (Untitled) Program Editor ... PROC DEB

start Microsoft Power... SAS

SAS

File View Tools Run Breakpoint Solutions Window Help

DEBUGGER LOG

```
Command ==>
DATA STEP Source Level Debugger

Stopped at line 465 column 1
> examine _all_
name =
repeat = .
type =
Pet_nm =
someday = .
_ERROR_ = 0
_N_ = 1
> st; ex _all_
Stepped to line 467 column 1
name =
repeat = .
type =
Pet_nm =
someday = .
_ERROR_ = 0
_N_ = 1
> st; ex _all_
Stepped to line 470 column 1
name = Art
repeat = 0.01
type = Dog
Pet_nm = Catcher
someday = 02JAN1960
_ERROR_ = 0
_N_ = 1
```

DEBUGGER SOURCE

```
Command ==>
464 data Prblm_pets (sortedby=name)/debug;
465 infile datalines missover firstobs=2;
466 *retain repeat;
467 input @2 name $char5. @9 repeat 4.2 @15
468                                @9 someday ;
469 format someday date9.;
470 datalines;
```

**Examine syntax is:**

**examine** { **\_all\_** **variable format** **var1 var2 var6** }

**<F4> till you see the last command and then ret**

**The EXAMINE command displays the value of one or more specified variables. The debugger displays the value using the format currently associated with the variable, unless you specify a different format.**

Output - (Untitl... Log - (Untitled) Program Editor ... PROC DEB

**SAS**

File View Tools Run Breakpoint Solutions Window Help

**DEBUGGER LOG**

Command ==>  
DATA STEP Source Level Debugger

Stopped at line 465 column 1  
> examine \_all\_  
name =  
repeat = .  
type =  
Pet\_nm =  
someday =  
\_ERROR\_ = 0  
\_N\_ = 1  
> st; ex \_all\_  
Stepped to line 467 column 1  
name =  
repeat = .  
type =  
Pet\_nm =  
someday =  
\_ERROR\_ = 0  
\_N\_ = 1  
> st; ex \_all\_  
Stepped to line 470 column 1  
name = Art  
repeat = 0.01  
type = Dog  
Pet\_nm = Catcher  
someday = 02JAN1960  
\_ERROR\_ = 0  
\_N\_ = 1  
> e name type  
name = Art  
type = Dog  
> e n 8.5  
Variable n is unknown  
> e \_n\_ 8.5  
\_N\_ = 1.00000

**DEBUGGER SOURCE**

Command ==>  
464 data Prblm\_pets (sortedby=name)/debug;  
465 infile datalines missover firstobs=2;  
466 \*retain repeat;  
467 input @2 name \$char5. @9 repeat 4.2 @15  
468 @9 someday ;  
469 format someday date9.;  
470 datalines;

**Examine syntax is:**

**examine** { **\_all\_**  
**E** **variable format**  
**var1 var2 var6**

**E \_N\_ 8.5**

**The EXAMINE command displays the value of one or more specified variables. The debugger displays the value using the format currently associated with the variable, unless you specify a different format.**

Output - (Untit... Log - (Untitled) Program Editor ... PROC DE

start Microsoft Power... SAS

SAS

File View Tools Run Breakpoint Solutions Window Help

DEBUGGER LOG

```
Command ==>
> st; ex _all_
Stepped to line 467 column 1
name =
repeat = .
type =
Pet_nm =
someday =
_ERROR_ = 0
_N_ = 1
> st; ex _all_
Stepped to line 470 column 1
name = Art
repeat = 0.01
type = Dog
Pet_nm = Catcher
someday = 02JAN1960
_ERROR_ = 0
_N_ = 1
> e name type
name = Art
type = Dog
> e n 8.5
Variable n is unknown
> e _n_ 8.5
_N_ = 1.000000
> list _all_
No breakpoints set
No watchpoints set
OUTPUT Datasets :
  Name = WORK.PRBLM_PETS, Engine = V9,
                                Number of Variables = 5
FILES :
  Logical name = LOG
  Column = 1, Line = 49, LL = 6
INFILES :
  Logical name = CARDS
  Column = 1, Line = 1, Length = 33
  Current buffer =
  Art      1      Dog  Catcher
>
```

DEBUGGER SOURCE

```
Command ==>
464 data Prblm_pets (sortedby=name)/debug;
465 infile datalines missover firstobs=2;
```

**List syntax is:**

- Use names or abbrevs
- \_all\_**
- I** or infiles
- F** or Files
- D** or datasets
- B** or breakpoints
- W** or watchpoints

**You can combine commands**

**List \_all\_**

**The LIST command displays all occurrences of the item specified in its argument.**

Output - (Untitl... Log - (Untitled) Program Editor ... PROC DEB

**SAS**

File View Tools Run Breakpoint Solutions Window Help

**DEBUGGER LOG**

```
Command ==>
someday =
_ERROR_ = 0
_N_ = 1
> st; ex _all_
Stepped to line 470 column 1
name = Art
repeat = 0.01
type = Dog
Pet_nm = Catcher
someday = 02JAN1960
_ERROR_ = 0
_N_ = 1
> e name type
name = Art
type = Dog
> e n 8.5
Variable n is unknown
> e _n_ 8.5
_N_ = 1.00000
> list _all_
No breakpoints set
No watchpoints set
OUTPUT Datasets :
  Name = WORK.PRBLM_PETS, Engine = V9,
                                Number of Variables = 5
FILES :
  Logical name = LOG
  Column = 1, Line = 49, LL = 6
INFILES :
  Logical name = CARDS
  Column = 11, Line = 1, Length = 80
  Current buffer =
    Art    1    Dog    Catcher
> list b d w
No breakpoints set
OUTPUT Datasets :
  Name = WORK.PRBLM_PETS, Engine = V9,
                                Number of Variables = 5
No watchpoints set
-----
> |
```

**DEBUGGER SOURCE**

```
Command ==>
464 data Prblm_pets (sortedby=name)/debug;
465 infile datalines missover firstobs=2;
```

**List syntax is:**

- Use names or abbrevs
- \_all\_**
- I** or infiles
- F** or Files
- D** or datasets
- B** or breakpoints
- W** or watchpoints

**You can combine commands**

**List B D W**

**The LIST command displays all occurrences of the item specified in its argument.**

Output - (Untitl... | Log - (Untitled) | Program Editor ... | PROC DEB



SAS

File View Tools Run Breakpoint Solutions Window Help

DEBUGGER LOG

```
Command ==>
someday = 02JAN1960
_ERROR_ = 0
_N_ = 1
> e name type
name = Art
type = Dog
> e n 8.5
Variable n is unknown
> e _n_ 8.5
_N_ = 1.00000
> list _all_
No breakpoints set
No watchpoints set
OUTPUT Datasets :
  Name = WORK.PRBLM_PETS, Engine = V9,
    Number of Variables = 5
FILES :
  Logical name = LOG
  Column = 1, Line = 49, LL = 6
INFILES :
  Logical name = CARDS
  Column = 11, Line = 1, Length = 80
  Current buffer =
    Art      1      Dog  Catcher
> list b d w
No breakpoints set
OUTPUT Datasets :
  Name = WORK.PRBLM_PETS, Engine = V9,
    Number of Variables = 5
No watchpoints set
> desc _all_
Name = name, Type = CHAR, Length = 5
Name = repeat, Type = NUM, Length = 8
Name = type, Type = CHAR, Length = 3
Name = Pet_nm, Type = CHAR, Length = 8
Name = someday, Type = NUM, Length = 8
Format = DATE9.
Name = _ERROR_, Type = NUM, Length = 8
Name = _N_, Type = NUM, Length = 8
>
```

DEBUGGER SOURCE

```
Command ==>
464 data Prblm_pets (sortedby=name)/debug;
465 infile datalines missover firstobs=2;
466 *retain repeat;
```

**Describe syntax is:**

**Describe** { **\_all\_**  
**DESC** var1 var2 ..

**Describe \_all\_**

**The DESCRIBE command displays the attributes of one or more specified variables.**

Output - (Untitl... Log - (Untitled) Program Editor ... PROC DEBUG

start Microsoft Power... SAS 6:02 PM

**SAS**

File View Tools Run Breakpoint Solutions Window Help

**DEBUGGER LOG**

Command ===>  
DATA STEP Source Level Debugger

Stopped at line 522 column 1  
> break \*  
Breakpoint 1 set at line 522

**DEBUGGER SOURCE**

```
521 data non_profit /debug;  
522 infile datalines missover firstobs=3;  
523 length age_grp 5;  
524 input @1 name $char8.  
525         @11 gender $char1.  
526         @13 age1 2.0  
527         @13 age2 2.0  
528         @17 From $char8.  
529         @25 Q1_hrs 3.  
530         @30 Q2_hrs 3.  
531         @35 region 1.;  
532 total= Q1_hrs + Q2_hrs;  
533 do i=1 to age1;  
534     stopper="here";  
535     *YOU CAN ALSO DIVIDE THE MAX BY AN INTEGER  
536     if i LE 12  
537         then age_grp = "Child";  
538     else if i GT 12 and i LT 20  
539         then age_grp = "Teen";  
540     else if i GE 21  
541         then age_grp = "Adult";  
542 end;  
543 pct_1= Q1_hrs /total;  
544 pct_2= Q1_hrs /total;  
545 datalines;
```

**Syntax for SETTING A BREAK POINT is:**

Break { location  
location <AFTER count>  
location <WHEN expression>  
location <DO group>

**Break \***

**The BREAK command “tags” a line on which we want the Data step to stop processing. Executing a BREAK , “tagging a line”, is usually called setting a breakpoint.**



**The BREAK command suspends execution of the DATA step at a specified statement. Executing BREAK is called setting a breakpoint.**

**Break syntax for SETTING A BREAK POINT is:**

**Break   location   <AFTER count>   < WHEN expression>   <DO group>**

**Break** { **Three ways to specify on which line to stop.**  
    **location** { \* (asterisk) - mark current line so DSD will stop  
                  Line Number - mark line so DSD will stop at line  
                  Label - mark line so DSD will stop at that label  
  
    **location <AFTER count>** - stop upon reaching line n times  
  
    **location < WHEN expr.>** stop on line when expression is true. Expressions can't contain functions.  
  
    **location <DO group>** one or more DSD commands in a DO -- END statement. **Note:** These commands will be executed when the debugger suspends program execution for the breakpoint.

The screenshot shows the SAS Debugger interface. The top menu bar includes File, View, Tools, Run, Breakpoint, Solutions, Window, and Help. Below the menu is a toolbar with various icons. The main window is divided into two panes: 'DEBUGGER LOG' on the left and 'DEBUGGER SOURCE' on the right. The 'DEBUGGER LOG' pane shows the command '====>' and the status 'DATA STEP Source Level Debugger'. It indicates that the debugger stopped at line 522, column 1, and that breakpoints were set at lines 522 and 532. The 'DEBUGGER SOURCE' pane displays the SAS code. Line 532, which is highlighted with a red box, contains the command '! 532 total= Q1\_hrs + Q2\_hrs;'. A red arrow points from this line to a red box at the bottom of the source pane that contains the text 'Break 532'. At the bottom of the window, there is a taskbar with the Windows Start button and several open applications, including Microsoft PowerPoint and the SAS application.

**Syntax for SETTING A BREAK POINT is:**

Break { location  
location <AFTER count>  
location <WHEN expression>  
location <DO group>

**Break 532**

**The BREAK command “tags” a line on which we want the Data step to stop processing. Executing a BREAK, “tagging a line”, is usually called setting a breakpoint.**

```
SAS
File View Tools Run Breakpoint Solutions Window Help

DEBUGGER LOG
Command ==>
> delete b _all_
All breakpoints cleared
> b 536 after 3; ex _all_;
Breakpoint 5 set at line 536
age_grp =
name =
gender =
age1 = .
age2 = .
From =
Q1_hrs = .
Q2_hrs = .
region = .
total = .
i = .
stopper =
pct_1 = .
pct_2 = .
_ERROR_ = 0
_N_ = 2
> go
Break at line 536 column 3
> ex _all_
age_grp = Child
name = Mike
gender =
age1 = 12
age2 = 12
From = Conn
Q1_hrs = 27
Q2_hrs = 42
region = 1
total = 69
i = 3
stopper = here
pct_1 = .
pct_2 = .
_ERROR_ = 0
_N_ = 2
>
```

## Syntax for SETTING A BREAK POINT is:

Break

location

location <AFTER count>

location <WHEN expression>

location <DO group>

```
531 do i=1 to age1;
532   stopper="here";
533   *YOU CAN ALSO DIVIDE THE MAX BY AN INTEGER /
536   if i LE 12
537     then age_grp = "Child";
538   else if i GT 12 and i LT 20
539     then age_grp = "Teen";
540   else if i GE 21
541     then age_grp = "Adult";
542 end;
543 pct_1= Q1_hrs /total;
544 pct_2= Q2_hrs /total;
545 datalines;
```

**B 536 after 3; ex \_all\_**

The **BREAK** command “tags” a line on which we want the Data step to stop processing. Executing a **BREAK**, “tagging a line”, is usually called **setting a breakpoint**.

```
DEBUGGER LOG
Command ==>
DATA STEP Source Level Debugger

Stopped at line 49 column 2
> break 52 when first.name*last.name = 0
Breakpoint 1 set at line 52
> go
Break at line 52 column 4
> ex _all_
h = 1
p = 1
name = Jen
time_w_co = 13
sex = F
job = Stat
line_no = 5
repeat = 1
type = Cat
Pet_nm = Princess
FIRST.name = 1
LAST.name = 0
stop_pt = .
_ERROR_ = 0
_N_ = 5
```

```
DEBUGGER SOURCE
Command ==>
48 data matching/debug;
49 merge humans(in=h) pets(in=p);
50 *stop_pt=1;
51 by name;
! 52 time_w_co=time_w_co*12;
53 stop_pt=1;
54 run;
```

**Syntax for SETTING A BREAK POINT is:**

Break {  
location  
location <AFTER count>  
location <WHEN expression>  
location <DO group>

**The BREAK command “tags” a line on which we want the Data step to stop processing. Executing a BREAK , “tagging a line”, is usually called **setting** a breakpoint.**

## DEBUGGER LOG

Command ==&gt;

DATA STEP Source Level Debugger

Stopped at line 181 column 1

&gt; B 193 DO; IF I IN(1,AGE1) THEN EX \_ALL; ELSE EX I AGE1; END;

Breakpoint 1 set at line 193

**Syntax for SETTING A BREAK POINT is:**

Break

**location****location <AFTER count>****location <WHEN expression>****location <DO group>**

## DEBUGGER SOURCE

Command ==&gt;

180 data non\_profit /debug;

181 infile datalines misover firstobs=3

182 length age\_grp \$ 5;

183 input @1 name \$char8.

**Set a break at the lowest and highest pass in the loop.**

189 180 Q2\_hrs 3.

190 181 region 1.;

191 182 total\_hrs = Q2\_hrs;

192 do i=1 to age1;

! 193 stopper="here";

194 \*YOU CAN ALSO DIVIDE THE MAX BY AN

195 if i LE 12

196 then age\_grp = "Child";

197 else if i GT 12 and i LT 20

198 then age\_grp = "Teen";

199 else if i GE 21

200 then age\_grp = "Adult";

201 end;

202 pct\_1= Q1\_hrs /total;

203 pct\_2= Q1\_hrs /total;

**B 193; DO IF I IN(1,AGE1) THEN EX \_ALL; ELSE EX I AGE1; END;**

**The BREAK command “tags” a line on which we want the Data step to stop processing. Executing a BREAK, “tagging a line”, is usually called setting a breakpoint.**

Since variables from datalines are re-set to missing at top of Data step,  
Add a retain  
to help with debugging.  
See line 296 in SAS code.

```
> go
Break at line 48 column 1
> examine all
```

ne 53  
'  
8

**Syntax for SETTING A Watch POINT is:**

**Watch Variable**

```
PctY1 = .
PctY2 = .
i = .
_ERROR_ = 0
_N_ = 3
```

```
> █
```

Output - (Untitled)

Log - (Untitled)



### DEBUGGER SOURCE

```
41 data show /DEBUG;
42 infile datalines missover;
43 input @2 name $char8.
44         @10 gender $char1.
45         @12 Region $char2.
46         @16 FY1 3.0
47         @20 FY2 3.0;
! 48 total=fy1+fy2;
49 array revenue(2) Fy1 Fy2;
50 array pcnts(2) PctY1 PctY2
51 do i=1 to 2;
52     pcnts(i)=Fy1/total;
53 end;
54 datalines;
```

**Watch Region**

**Go (no example shown)**

**Watch is useful for checking when “flags are raised” and in many other uses. Since you are asking the DSD to watch a variable and break when the value changes, you do not specify a “stopping location” or line number.**

## Syntax for SETTING Trace is

Trace { On  
Off

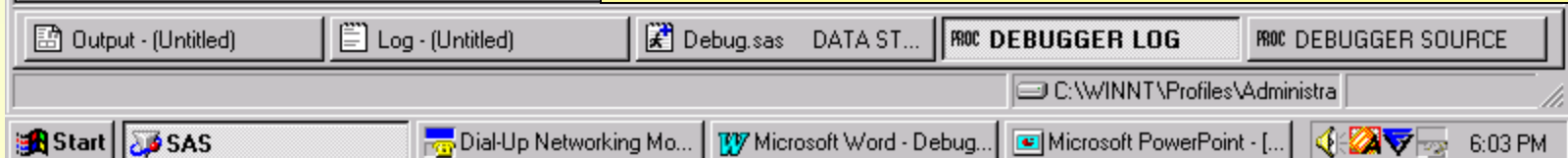
```
> D B 53
Breakpoint 1 cleared at line 53
> BREAK 48 WHEN GENDER = ''
Breakpoint 6 set at line 48
> go
Break at line 48 column 1
> examine _all_
name = rin
gender =
Region = SW
FY1 = .
FY2 = -10
total = .
PctY1 = .
PctY2 = .
i = .
_ERROR_ = 0
_N_ = 3
```

```
41 data show /DEBUG;
42 infile datalines missover;
43 input @2 name $char8.
44         @10 gender $char1.
45         @12 Region $char2.
46         @16 FY1 3.0
47         @20 FY2 3.0;
! 48 total=fy1+fy2;
49 array revenue(2) Fy1 Fy2;
50 array pcnts(2) PctY1 PctY2
51 do i=1 to 2;
52     pcnts(i)=Fy1/total;
53 end;
54 datalines;
```

Trace

(no example shown)

Trace writes additional material to the  
debugger log.





**Syntax for SWAP is :**

**SWAP**

```
name = rln  
gender =  
Region = SW  
FY1 = .  
FY2 = -10  
total = .  
PctY1 = .  
PctY2 = .  
i = .  
_ERROR_ = 0  
_N_ = 3  
> swap
```

**Active window changes**

**DEBUGGER SOURCE**

```
41 data show /DEBUG;  
42 infile datalines missover;  
43 input @2 name $char8.  
44         @10 gender $char1.  
45         @12 Region $char2.  
46         @16 FY1 3.0  
47         @20 FY2 3.0;  
! 48 total=fy1+fy2;  
49 array revenue(2) Fy1 Fy2;  
50 array pcnts(2) PctY1 PctY2  
51 do i=1 to 2;  
52     pcnts(i)=Fy1/total;  
53 end;  
54 datalines;
```

**SWAP**

**The SWAP command switches control between the LOG window and the SOURCE so that you can scroll and view the text of the program. Issue from command line or menu. (also click mouse)**

**Syntax for CALCULATE is:**

**Calculate Expression**

```
> examine _OUT_
name = rin
gender =
Region = SW
FY1 = .
FY2 = -10
total = .
PctY1 = .
PctY2 = .
i = .
_ERROR_ = 0
_N_ = 3
> calculate fy2*2
-20
```

**-20**

```
41 data show /DEBUG;
42 infile datalines missover;
43 input @2 name $char8.
44         @10 gender $char1.
45         @12 Region $char2.
46         @16 FY1 3.0
47         @20 FY2 3.0;
! 48 total=fy1+fy2;
49 array revenue(2) Fy1 Fy2;
50 array pcnts(2) PctY1 PctY2
51 do i=1 to 2;
52     pcnts(i)=Fy1/total;
53 end;
54 datalines;
```

**Calculate FY2\*2**

**The CALCULATE command evaluates debugger expressions and displays the result. Debugger expressions cannot contain functions.**

**Syntax for CALCULATE is:**

**Calculate Expression**

```
> examine _OUT_
name = rin
gender =
Region = SW
FY1 = .
FY2 = -10
total = .
PctY1 = .
PctY2 = .
i = .
_ERROR_ = 0
_N_ = 3
> calc fy1+fy2
.
```

**. (missing value)**

```
41 data show /DEBUG;
42 infile datalines missover;
43 input @2 name $char8.
44         @10 gender $char1.
45         @12 Region $char2.
46         @16 FY1 3.0
47         @20 FY2 3.0;
! 48 total=fy1+fy2;
49 array revenue(2) Fy1 Fy2;
50 array pcnts(2) PctY1 PctY2
51 do i=1 to 2;
52     pcnts(i)=Fy1/total;
53 end;
54 datalines;
```

**Calculate FY1+FY2**

**The CALCULATE command evaluates debugger expressions and displays the result. Debugger expressions cannot contain functions.**

**Syntax for CALCULATE is:**

**Calculate Expression**

```
> examine _OUT_
name = rin
gender =
Region = SW
FY1 = .
FY2 = -10
total = .
PctY1 = .
PctY2 = .
i = .
_ERROR_ = 0
_N_ = 3
> calc sum(fy1,fy2)
Variable sum is unknown
```

**Variable sum is unknown**

```
41 data show /DEBUG;
42 infile datalines missover;
43 input @2 name $char8.
44         @10 gender $char1.
45         @12 Region $char2.
46         @16 FY1 3.0
47         @20 FY2 3.0;
! 48 total=fy1+fy2;
49 array revenue(2) Fy1 Fy2;
50 array pcnts(2) PctY1 PctY2
51 do i=1 to 2;
52     pcnts(i)=Fy1/total;
53 end;
54 datalines;
```

**Calculate Sum(fy1,fy2)**

**The CALCULATE command evaluates debugger expressions and displays the result. Debugger expressions cannot contain functions.**

## Syntax for CALCULATE is:

Calculate Expression

```
gender = SW
Region = SW
FY1 = .
FY2 = -10
total = .
PctY1 = .
PctY2 = .
i = .
_ERROR_ = 0
_N_ = 3
> calc fy1+fy2
.
> calc Gender||region
Invalid numeric expression
-----
>
```

Invalid numeric expression

```
DEBUGGER SOURCE
41 data show /DEBUG;
42 infile datalines missover;
43 input @2 name $char8.
44         @10 gender $char1.
45         @12 Region $char2.
46         @16 FY1 3.0
47         @20 FY2 3.0;
! 48 total=fy1+fy2;
49 array revenue(2) Fy1 Fy2;
50 array pcnts(2) PctY1 PctY2
51 do i=1 to 2;
52     pcnts(i)=Fy1/total;
53 end;
54 datalines;
```

Calculate Gender||Region

**The CALCULATE command evaluates debugger expressions and displays the result. Debugger expressions cannot contain functions.**

## Syntax for SET is:

**SET Variable=Expression**

```
_ERROR_ = 0
_N_ = 3
> examine _all_
name = rin
gender =
Region = SW
FY1 = .
FY2 = -10
total = .
PctY1 = .
PctY2 = .
i = .
_ERROR_ = 0
_N_ = 3
> calc sum(fy1,fy2)
Variable sum is unknown
> set gender="F"
>
```

**Confirmation of the set**  
**set gender="F"**

```
41 data show /DEBUG;
42 infile datalines missover;
43 input @2 name $char8.
44         @10 gender $char1.
45         @12 Region $char2.
46         @16 FY1 3.0
47         @20 FY2 3.0;
! 48 total=fy1+fy2;
49 array revenue(2) Fy1 Fy2;
50 array pcnts(2) PctY1 PctY2
51 do i=1 to 2;
52     pcnts(i)=Fy1/total;
53 end;
54 datalines;
```

**Set Gender="F"**

**The SET command assigns a value to a specified variable. When you detect an error during program execution, you can use this command to assign new values to variables.**

**Syntax for SET is:**

**SET Variable=Expression**

```
_ERROR_ = 0
_N_ = 3
> examine _all_
name = rin
gender =
Region = SW
FY1 = .
FY2 = -10
total = .
PctY1 = .
PctY2 = .
i = .
_ERROR_ = 0
_N_ = 3
> calc sum(fy1,fy2)
Variable sum is unknown
> set gender="F"
> set i=5
> █
```

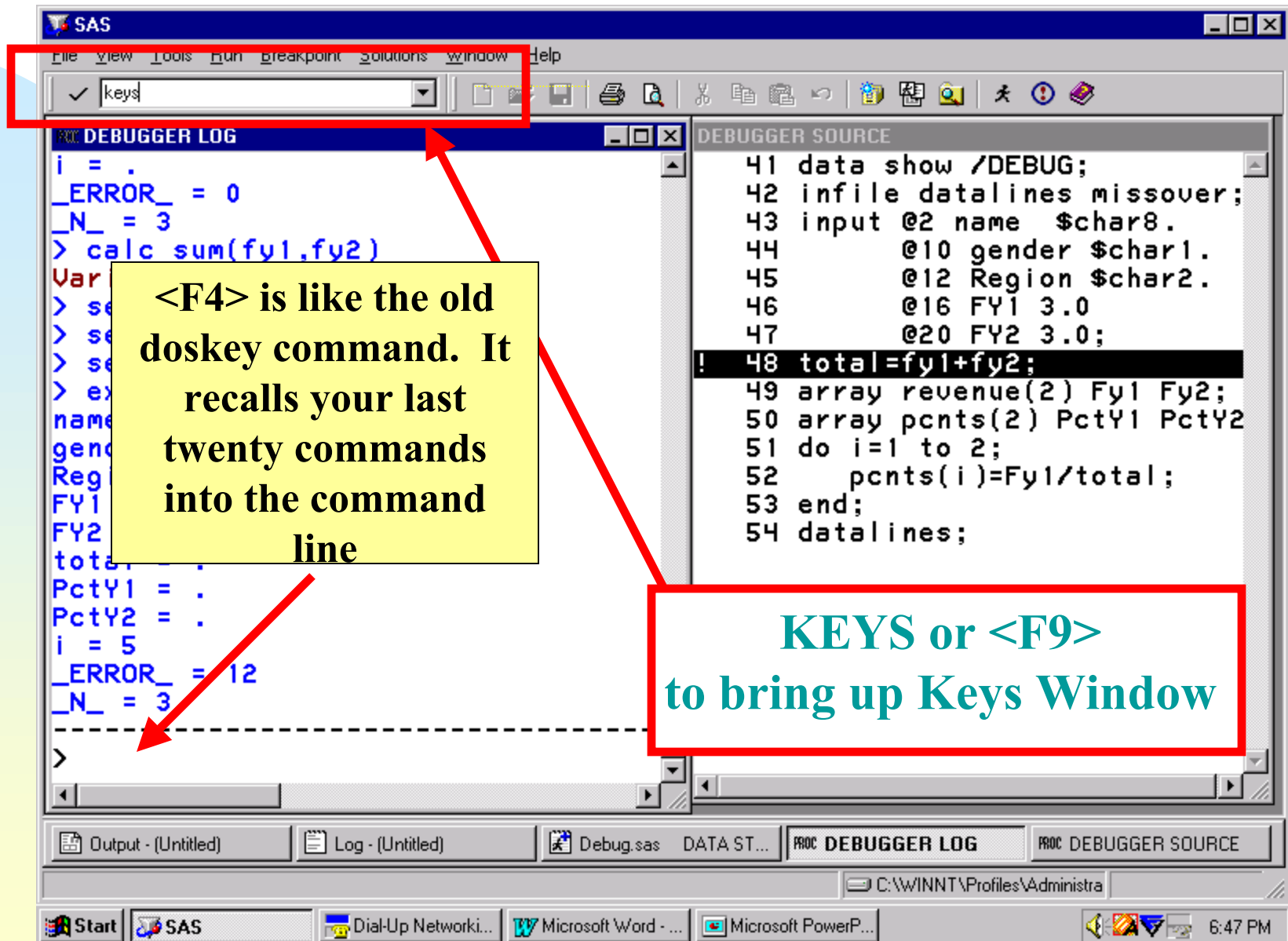
**Note you can set I out of range**

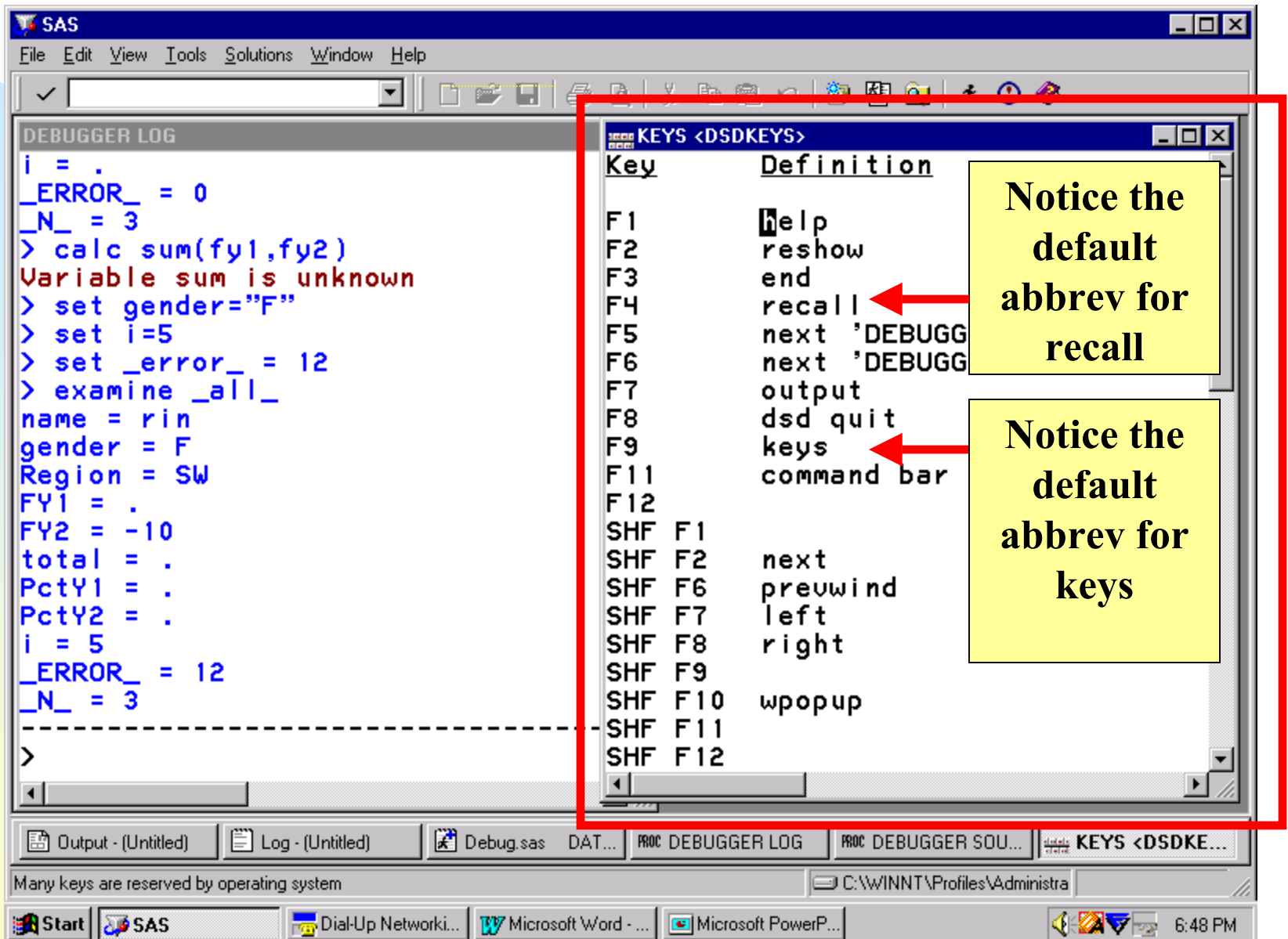
```
41 data show /DEBUG;
42 infile datalines missover;
43 input @2 name $char8.
44         @10 gender $char1.
45         @12 Region $char2.
46         @16 FY1 3.0
47         @20 FY2 3.0;
! 48 total=fy1+fy2;
49 array revenue(2) Fy1 Fy2;
50 array pcnts(2) PctY1 PctY2
51 do i=1 to 2;
52     pcnts(i)=Fy1/total;
53 end;
54 datalines;
```

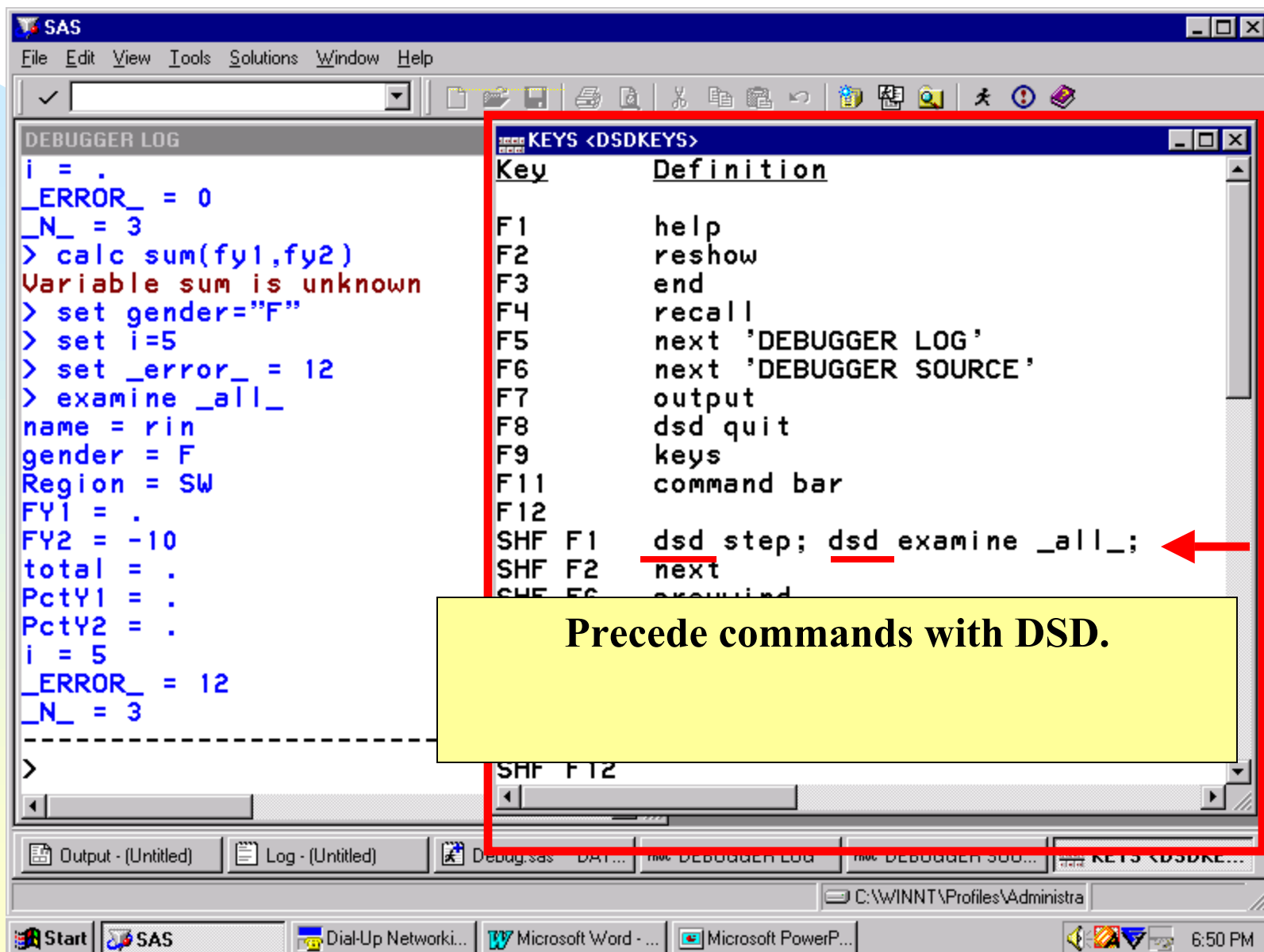
**Set I = 5**

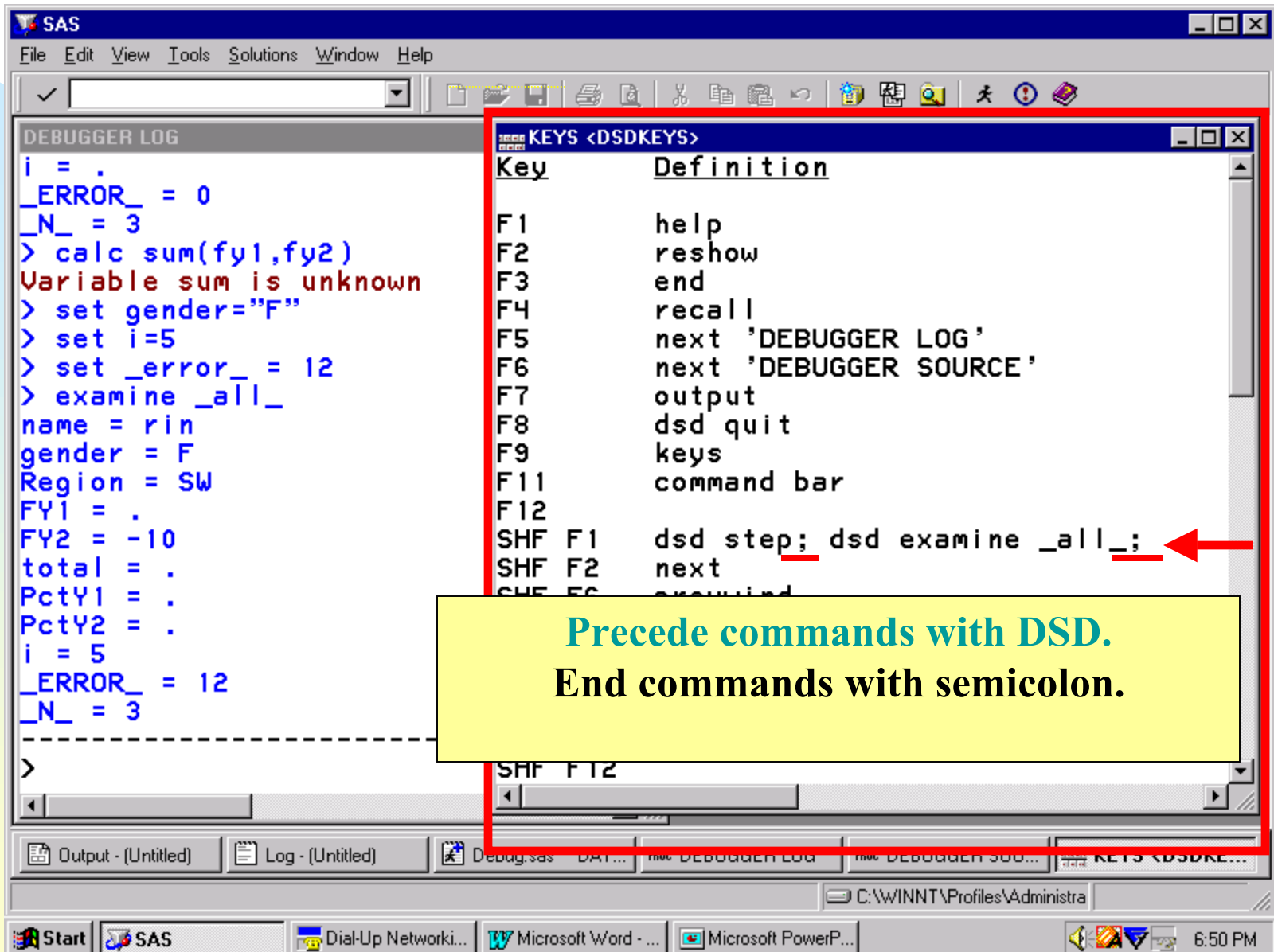
**The SET command assigns a value to a specified variable. When you detect an error during program execution, you can use this command to assign new values to variables.**

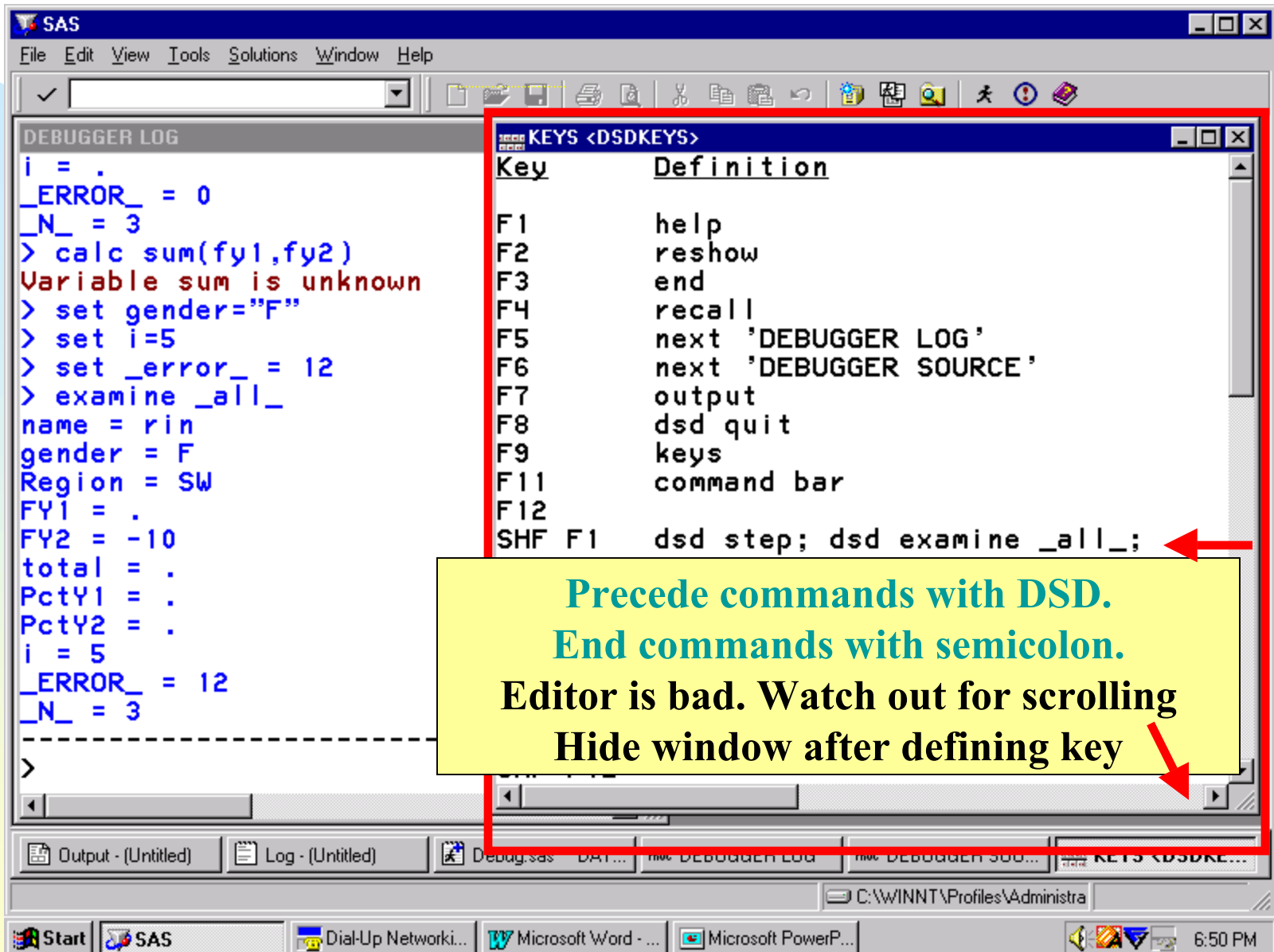


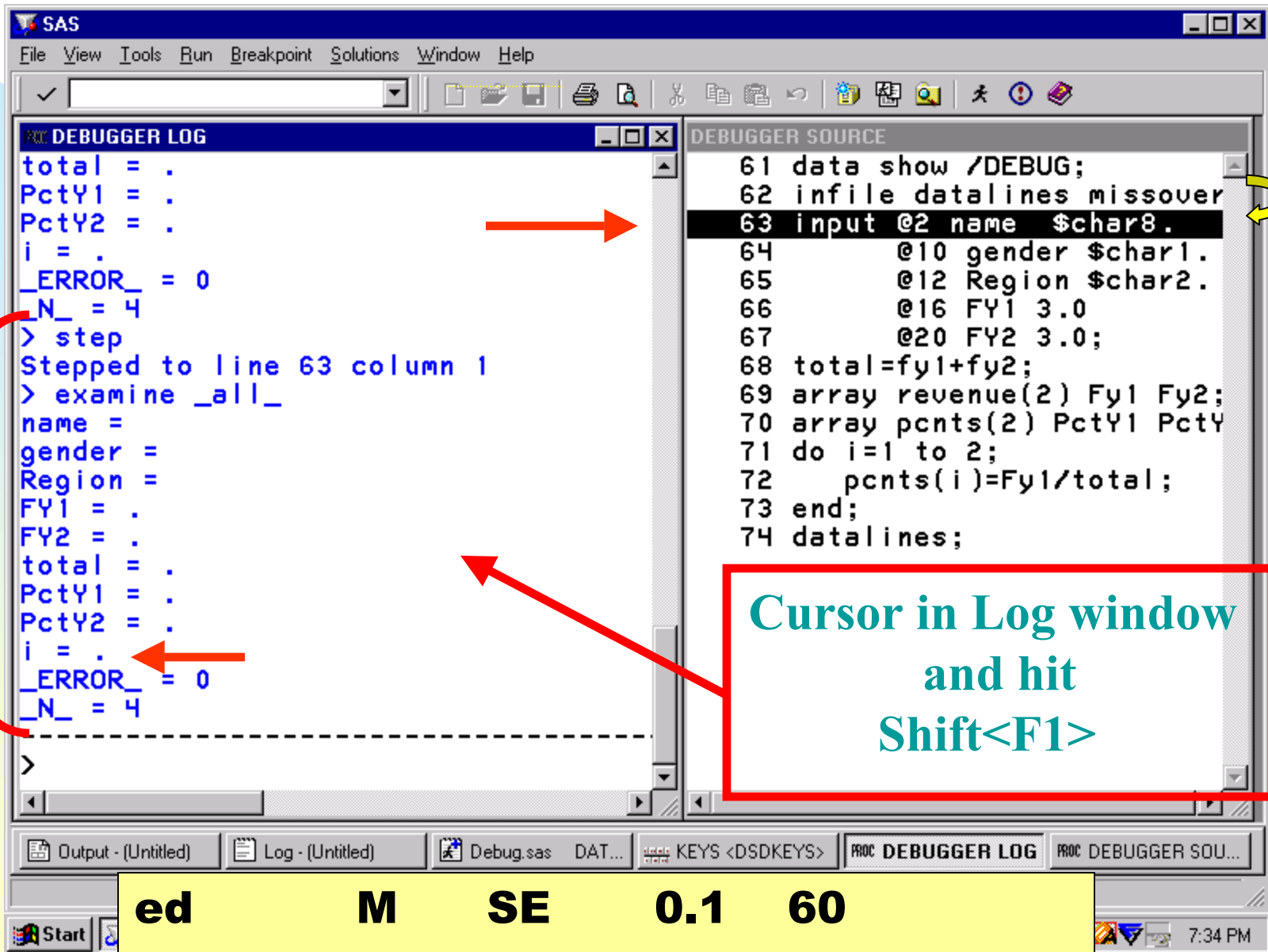


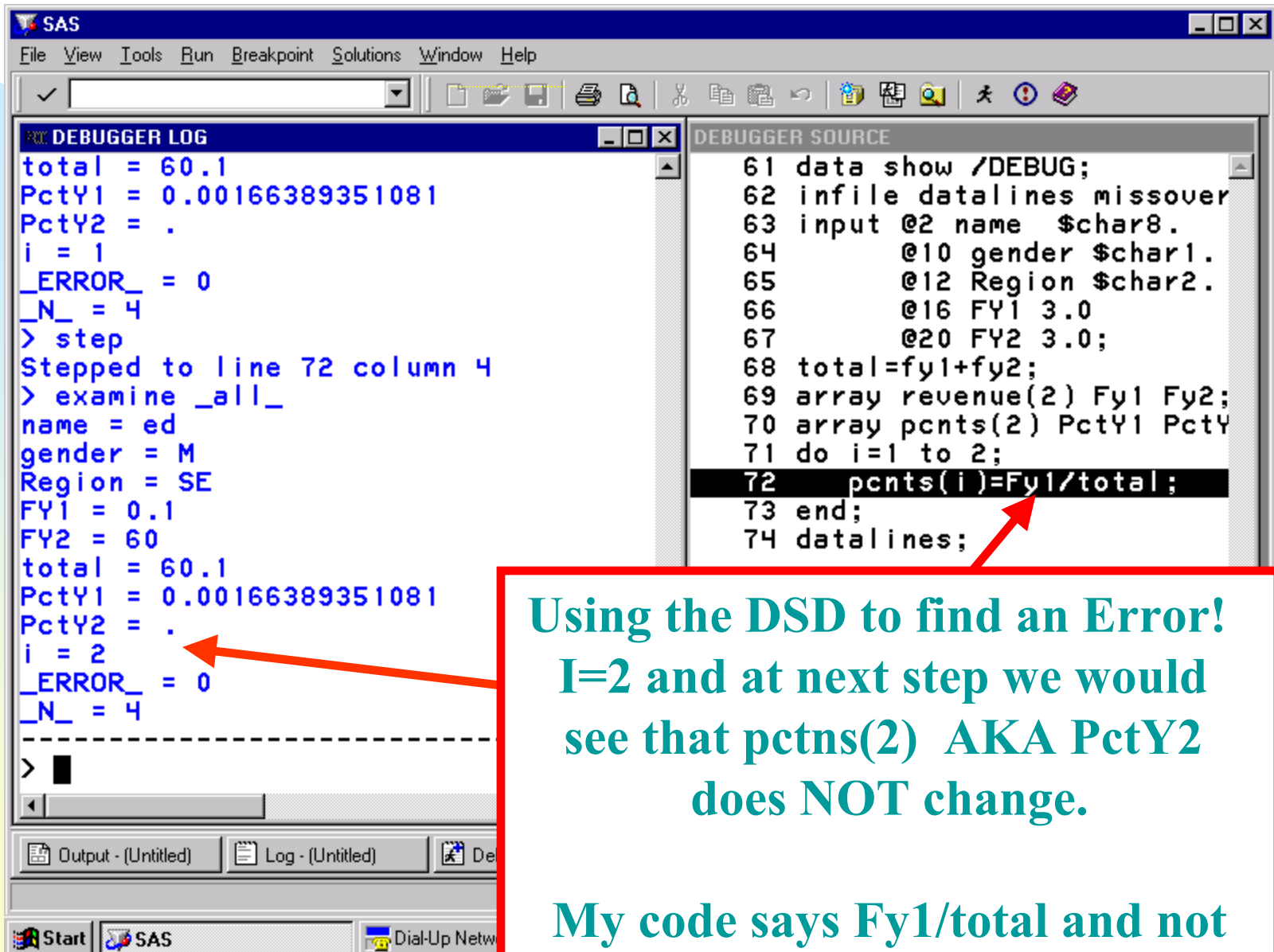












**Using the DSD to find an Error!**  
**I=2 and at next step we would see that pcnts(2) AKA PctY2 does NOT change.**

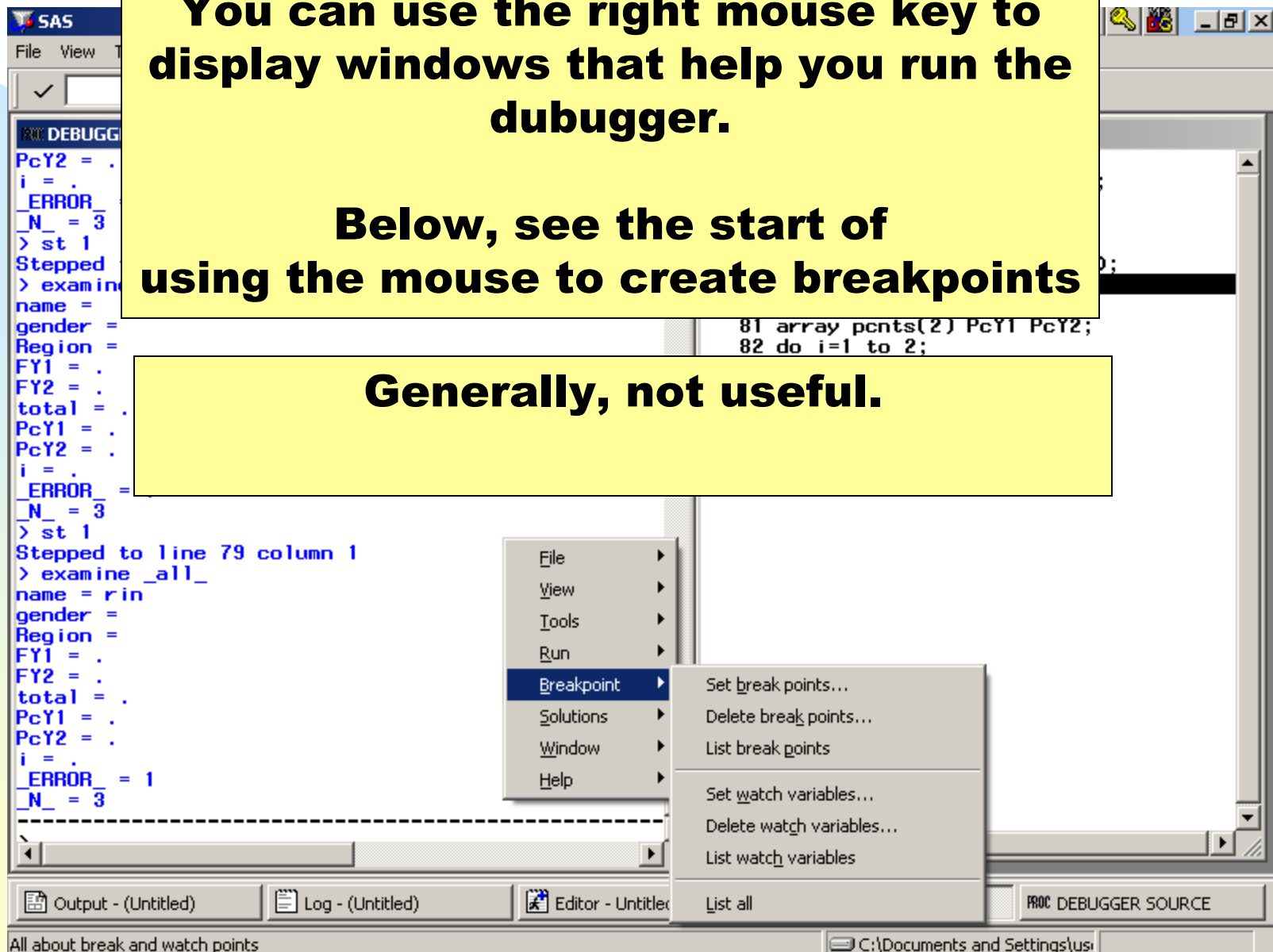
**My code says Fy1/total and not revenue(i)/total**



**You can use the right mouse key to display windows that help you run the debugger.**

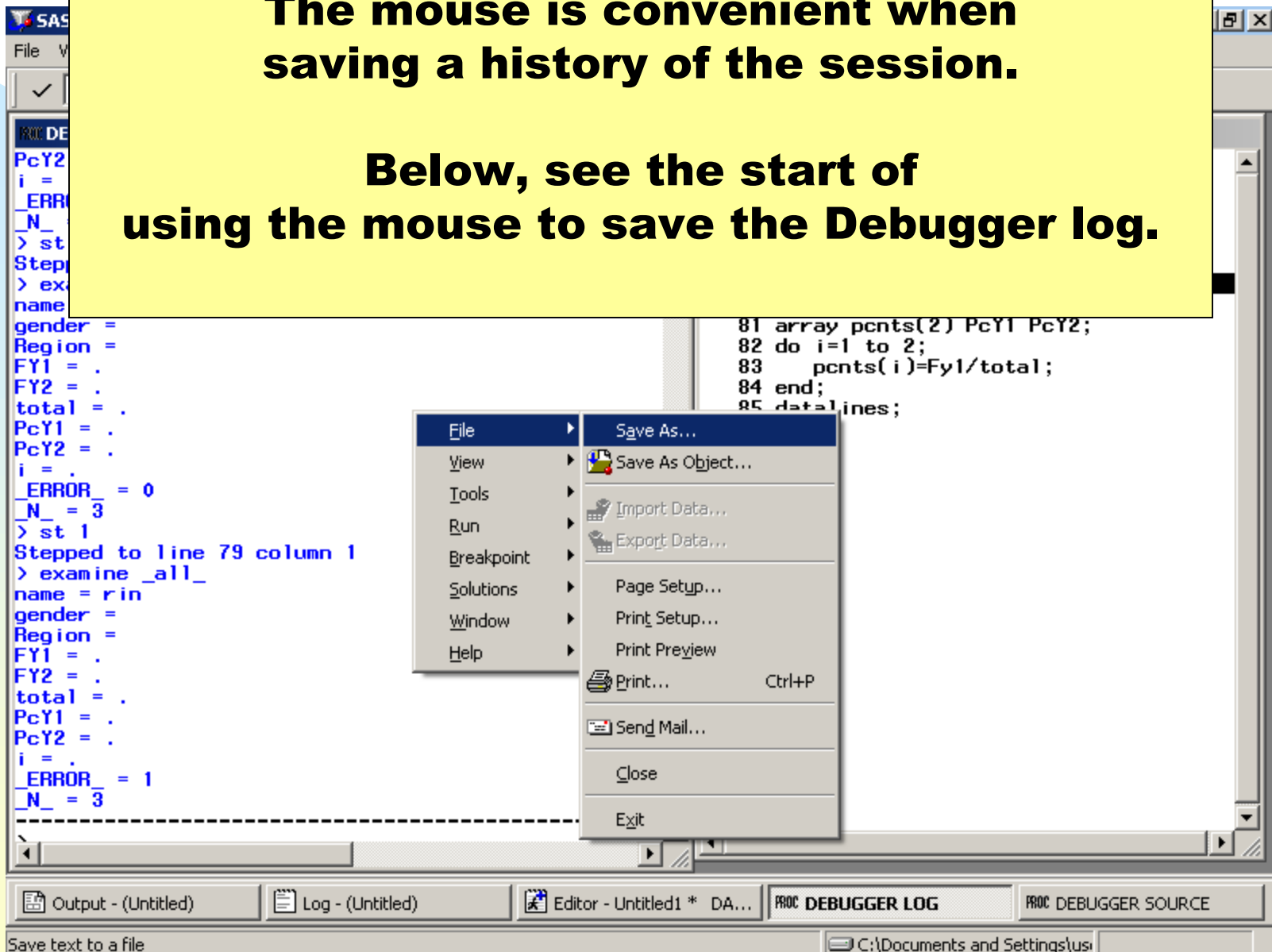
**Below, see the start of using the mouse to create breakpoints**

**Generally, not useful.**



**The mouse is convenient when saving a history of the session.**

**Below, see the start of using the mouse to save the Debugger log.**



**DATA SET**

**INPUT BUFFER**

**DATA VECTOR**

**INPUT STACK**

**WORD  
QUEUE**

**COMPILER**  
(SAS COMPILATION)

**EXECUTE**  
(SAS EXECUTION)

HOW  
HOW

**%INCLUDE**

**MACRO CATALOG**

**CALL EXECUTE**

**MULTICOMPONENT**

**Compile**

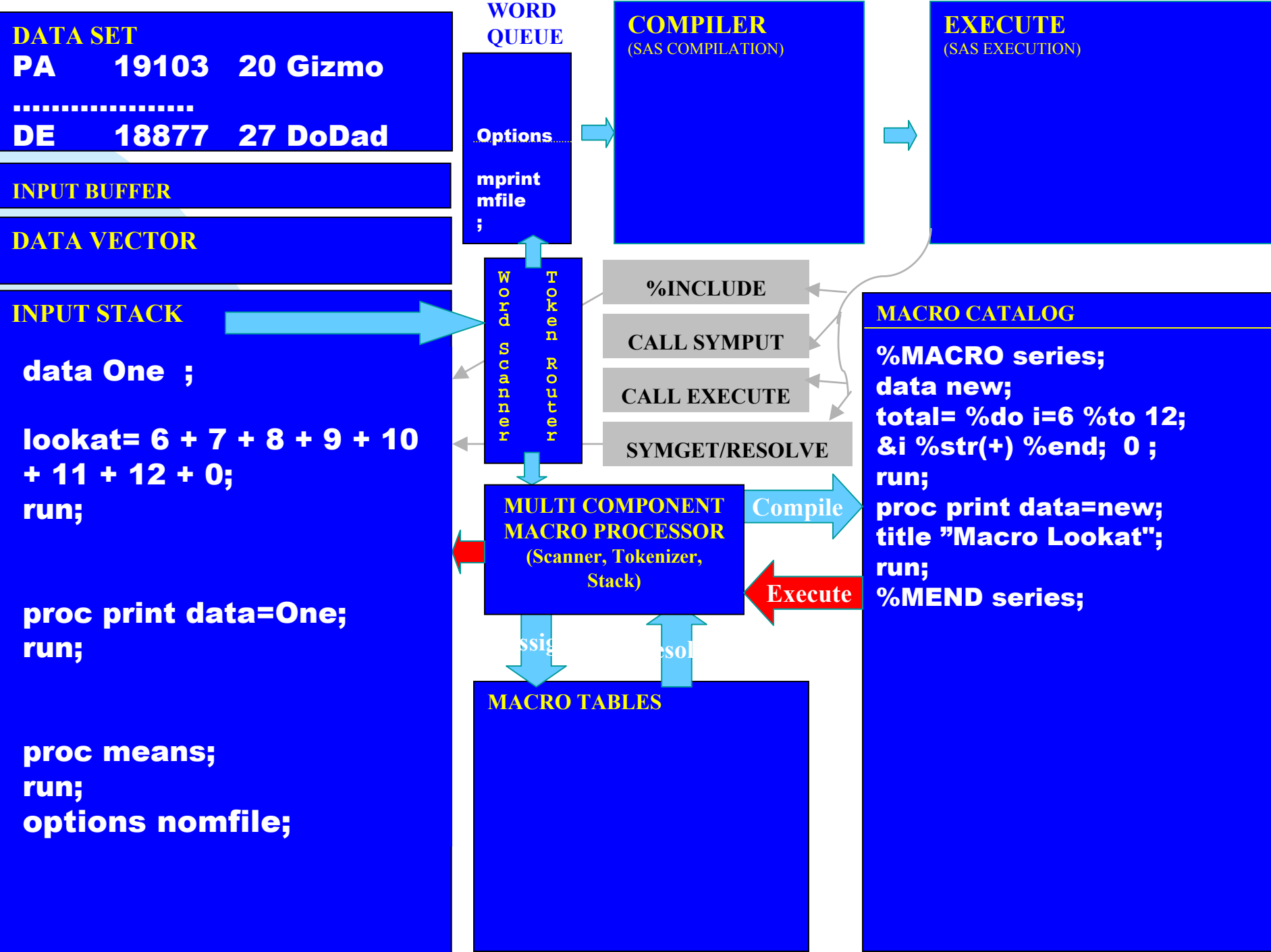
**Can we use the DSD with Macros??**

**Sort of!**

**The DSD will not help with the macro  
syntax (the %if... %then; stuff)**

**If the issue is associated with the code  
that the macro generated, you can save  
the output from macro execution and use  
the DSD to analyse the resulting code.**





```
%let dset=one; %let low=6; %let high=12;
```

```
%macro series;
```

```
  data &dset;
```

```
    lookat= %do i= &low %to &high;
```

```
      &i +
```

```
      %end;
```

```
  0;
```

```
  run;
```

```
  proc print data=&dset;
```

```
  run;
```

```
  proc means;
```

```
  run;
```

```
%mend series;
```

```
options mprint mfile;
```

```
filename mprint 'c:\temp\capture_code2.txt';  
%series;
```

```
options nomfile;
```

Hi Russell,  
Sorry to let you know that there is NOT a way to NOT append the mprint/mfile output.  
You will have to delete the file.  
Regards,  
-Scott @ SAS

```
options mprint mfile;
```

```
filename mprint 'c:\temp\capture_code2.txt';
```

```
  ***MACRO CALL HERE;
```

```
options nomfile;
```

**%let dset=one; %let low=6; %let high=12; russ.lavery@verizon.net**

**data One /DEBUG ;**

**lookat= 6 + 7 + 8 + 9 + 10 + 11 + 12 + 0;**

**run;**

**proc print data=One;**

**run;**

Hi Russell,  
Sorry to let you know that there is not a way to  
NOT append the mprint/mfile output.  
You will have to delete the file.

Regards,  
-Scott @ SAS

**proc means;**

**run;**

**options mprint mfile; \*for V7 and up;  
filename mprint 'c:\temp\lookSAS.txt';**

**Options nomprint;**

**The End**  
**Russell.Lavery@att.net**

**Thanks to:**

**Dr. Steve Bajgier**

**Skilled Reviewers:**  
**John Maurer, Saad Anbari, Musa Nsereko**

**See:**  
**How to use the data step Debugger SUGI 25 paper 52**  
**by Riba**