



Science For A Better Life

Macumba

State of the Art SAS GUI

Debugging made easy



Macumba – Debugging made easy – Overview

1. What is MACUMBA
 - Short Overview of the Application
2. Special SAS Code Execution Modes
 - Run (To/From) Line, Run Step
 - Run in Template, Run As Macro
3. SAS DATA-Step Debugger
 - Overview
 - Compare with Interactive SAS Debugger
4. SAS MACRO Debugger
 - Overview
 - Implementation
 - Outlook

Macumba – What is MACUMBA?



- In-House developed SAS IDE
- “Pseudo”-Interactive code execution on Remote SAS Server
- Focus on **development** of SAS programs & macros
- SAS code editor
 - Syntax Highlight
 - Code Completion, ...
- Data Set Viewer / Editor
 - Sorting, Filtering, Statistics, ...
 - Display of Formats / Labels

MACUMBA v1.5-20120521 - ToscaProductiveServer

Command: Enter Command

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

```
PROC FORMAT;  
  value $ _sex 'M' = 'Male' 'F' = 'Female';  
  picture _age 0-99 = '99 years';  
RUN;  
  
DATA class;  
  SET sashelp.class;  
  ATTRIB name LABEL = "Persons Name";  
  ATTRIB sex FORMAT = $ _sex. LABEL = "Persons Sex";  
  ATTRIB age FORMAT = _age. LABEL = "Persons Age";  
  ATTRIB height LABEL = "Persons Height";  
  ATTRIB weight LABEL = "Persons Weight";  
RUN;
```

Log

```
10 ATTRIB sex FORMAT = $ _sex. LA  
11 ATTRIB age FORMAT = _age. LA  
12 ATTRIB height LABEL = "Persons Height";  
13 ATTRIB weight LABEL = "Persons Weight";  
14 RUN;  
  
NOTE: There were 19 observations read from t  
NOTE: The data set WORK.CLASS has 19 observa  
NOTE: DATA statement used (Total process tim  
real time 0.00 seconds  
cpu time 0.01 seconds
```

Output

Name	Sex	Age	Height	Weight
Persons Name	Persons Sex	Persons Age	Persons Height	Persons Weight
1 Alfred	M - Male	14 - 14 years	69	112.5
2 Alice	F - Female	13 - 13 years	56.5	84
3 Barbara	F - Female	13 - 13 years	65.3	98
4 Carol	F - Female	14 - 14 years	62.8	102.6
5 Henry	M - Male	14 - 14 years	63.5	102.6
6 James	M - Male	12 - 12 years	57.3	83
7 Jane	F - Female	12 - 12 years	59.8	84.5
8 Janet	F - Female	15 - 15 years	62.5	112.5
9 Jeffrey	M - Male	13 - 13 years	62.5	84
10 John	M - Male	12 - 12 years	59	98.5
11 Joyce	F - Female	11 - 11 years	51.3	50.5
12 Judy	F - Female	14 - 14 years	64.3	90
13 Louise	F - Female	12 - 12 years	66.3	77
14 Mary	F - Female	16 - 16 years	66.5	112
15 Philip	M - Male	16 - 16 years	72	150
16 Robert	M - Male	12 - 12 years	64.8	128

14 lines of code executed successful in 78ms

Line: 6 Column: 1

Macumba – Special SAS Code Execution Modes 1

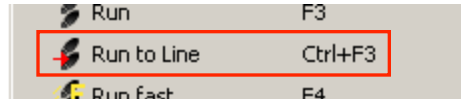


- PC-SAS provides two Execution Modes
 - Execute Selection (if code selected)
 - Execute All (if nothing selected)
- Uncomfortable when developing / debugging big programs
 - Execute from start to cursor position → scroll up / down
 - Execute from cursor position to program end → scroll down / up
 - ERROR when executing macro parts (%IF, %DO, %LOCAL, ...)

Macumba – Special SAS Code Execution Modes 2



- Run To Line



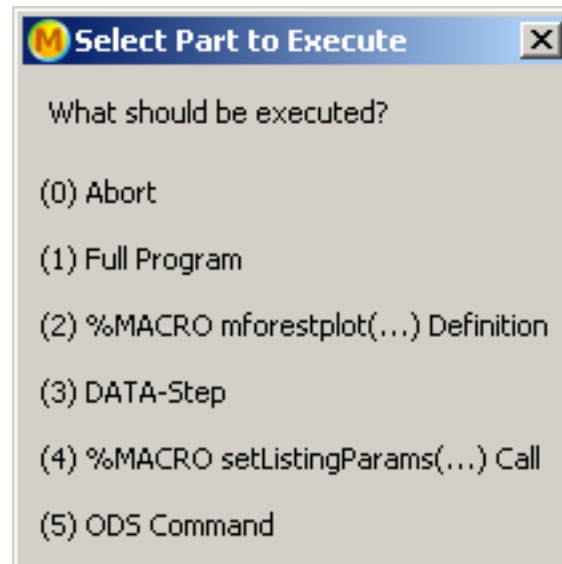
- Execute all code from start of program to cursor line

- Run From Line

- Execute all code from cursor line to end of program

- Run Step

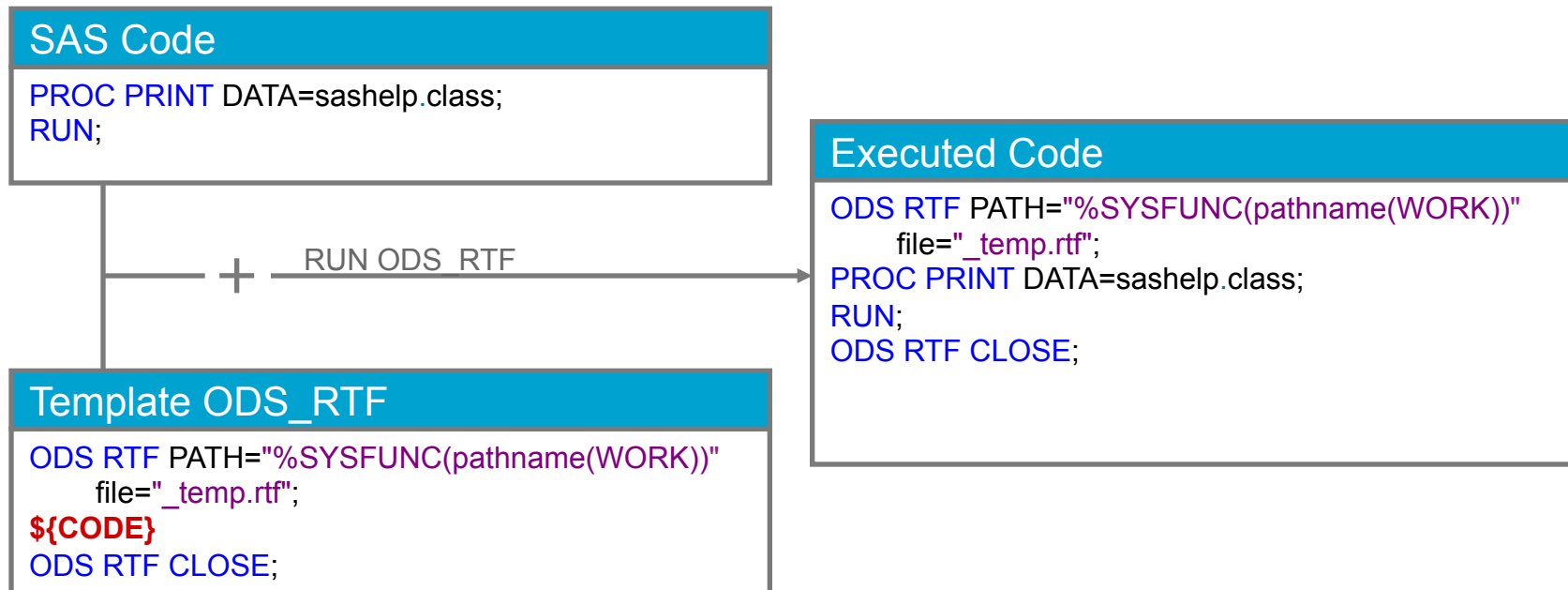
- Execute current step
 - %MACRO Definition
 - DATA- or PROC-Step
 - Global Command
 - Macro Call



Macumba – Special SAS Code Execution Modes 3



- Run in Template
 - Execute the code wrapped in the specified template
 - Wrap a PROC PRINT into ODS RTF command



Macumba – Special SAS Code Execution Modes 4



- Run As Macro

SAS Code

```
%MACRO abc(param1=hello, param2=world, max=10);  
  %LOCAL i;  
  %DO i=1 %TO &max.;  
    %IF %SYSFUNC(mod(&i., 2)) EQ 0  
    %THEN %DO;  
      %PUT &param1. &param2. (i=&i.);  
    %END;  
  %END;  
%MEND;  
%abc();
```

SAS Log

```
1      %abc();  
hello world (i=2)  
hello world (i=4)  
hello world (i=6)  
hello world (i=8)  
hello world (i=10)
```

What if (for debugging) only the %DO loop should be executed?

SAS Code

```
%DO i=1 %TO &max.;  
  %IF %SYSFUNC(mod(&i., 2)) EQ 0  
  %THEN %DO;  
    %PUT &param1. &param2. (i=&i.);  
  %END;  
%END;
```

SAS Log

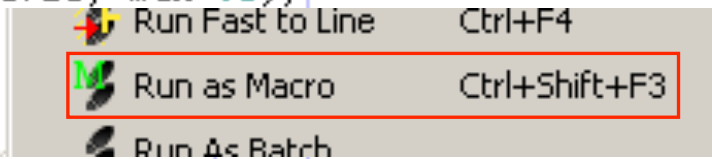
```
ERROR: The %DO statement is not valid in open  
code.  
ERROR: The %IF statement is not valid in open  
code.  
WARNING: Apparent symbolic reference PARAM1  
not resolved.  
WARNING: Apparent symbolic reference PARAM2  
not resolved.  
WARNING: Apparent symbolic reference I not  
resolved.
```

Macumba – Special SAS Code Execution Modes 4



- Run As Macro
 - MACUMBA action „Run As Macro“

```
2  %MACRO abc(param1=hello, param2=world, max=10);
3
4      %LOCAL i;
5  %DO i=1 %TO &max.;
6      %IF %SYSFUNC(mod(&i., 2)) EQ 0
7  %THEN %DO;
8          %PUT &param1. &param2. (i=&i.);
9      %END;
10 %END;
11
12 %MEND;
13 %abc();
```



Macumba – Special SAS Code Execution Modes 4



- Run As Macro
 - MACUMBA action „Run As Macro“

The screenshot illustrates the 'Run As Macro' action in the SAS IDE. A dialog box titled 'Select parameters to initialize' is open, showing three parameters: param1 (value: hello), param2 (value: world), and max (value: 10). The background shows a SAS code editor with a macro definition. A separate window titled 'SAS Code' displays the generated SAS code for this action.

Select parameters to initialize

Parameter	Value
☒ param1	hello
☒ param2	world
☒ max	10

SAS Code

```
%MACRO __MACUMBA_RUN_AS_MACRO;  
  %LOCAL param1 param2 max;  
  %LET param1 = hello;  
  %LET param2 = world;  
  %LET max = 10;  
  %DO i=1 %TO &max.;  
    %IF %SYSFUNC(mod(&i., 2)) EQ 0  
    %THEN %DO;  
      %PUT &param1. &param2. (i=&i.);  
    %END;  
  %END;  
%MEND __MACUMBA_RUN_AS_MACRO;  
%__MACUMBA_RUN_AS_MACRO;
```


Macumba – Special SAS Code Execution Modes 5



- Resolve Code
 - What SAS code is executed for:

SAS Code

```
%doStudyEvaluation(studyId=0815, projectId=4711);
```

- Execute:

SAS Code

```
OPTIONS MPRINT;  
%doStudyEvaluation(studyId=0815, projectId=4711);  
OPTIONS NOMPRINT;
```

- Check your log

SAS Log

```
1  OPTIONS MPRINT;  
2  %doStudyEvaluation(studyId=0815, projectId=4711);  
Will create study formats for 4711 / 0815!  
MPRINT(DOSTUDYEVALUATION):  PROC FORMAT;  
MPRINT(DOSTUDYEVALUATION):  PICTURE _age low-high = '99 Years';  
NOTE: Format _AGE is already on the library.  
NOTE: Format _AGE has been output.  
MPRINT(DOSTUDYEVALUATION):  RUN;  
  
NOTE: PROCEDURE FORMAT used (Total process time):  
      real time      0.00 seconds
```

Macumba – Special SAS Code Execution Modes 5



- Resolve Code

- Execute:

SAS Code

```
FILENAME mprint "%SYSFUNC(pathname(WORK))/resolved.sas";  
OPTIONS MPRINT MFILE;  
%doStudyEvaluation(studyId=0815, projectId=4711);  
OPTIONS NOMPRINT NOMFILE;  
FILENAME mprint;
```

- Check resolved.sas in work directory

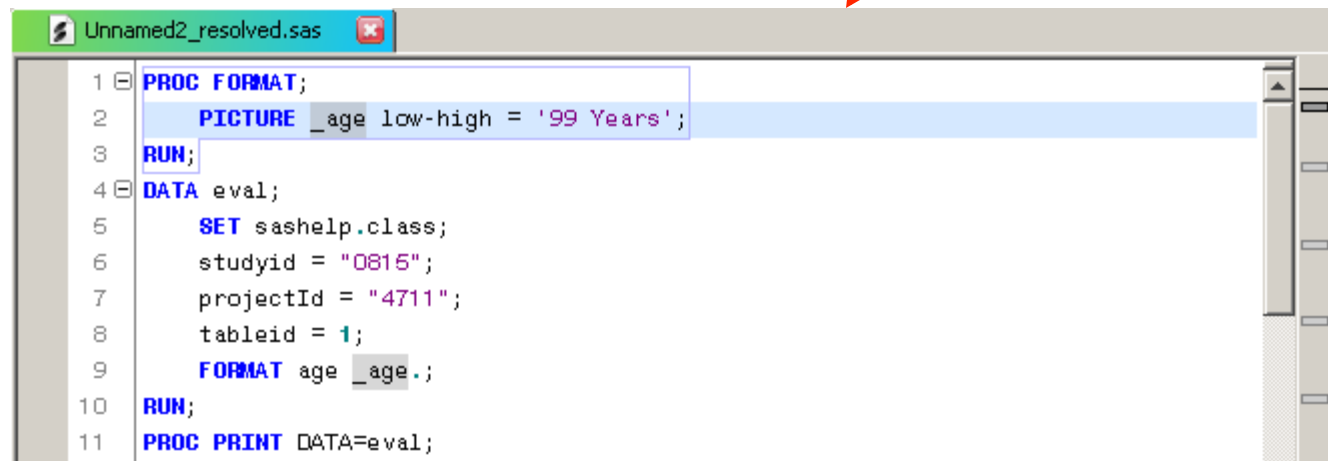
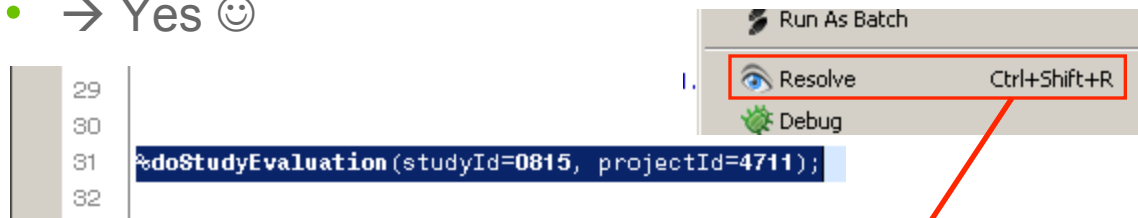
SAS Code (resolved.sas)

```
PROC FORMAT;  
PICTURE _age low-high = '99 Years';  
RUN;  
DATA eval;  
SET sashelp.class;  
studyid = "0815";  
projectId = "4711";  
tableid = 1;  
FORMAT age _age.;  
RUN;  
PROC PRINT DATA=eval;  
RUN;  
DATA eval;  
SET sashelp.class;
```

Macumba – Special SAS Code Execution Modes 5



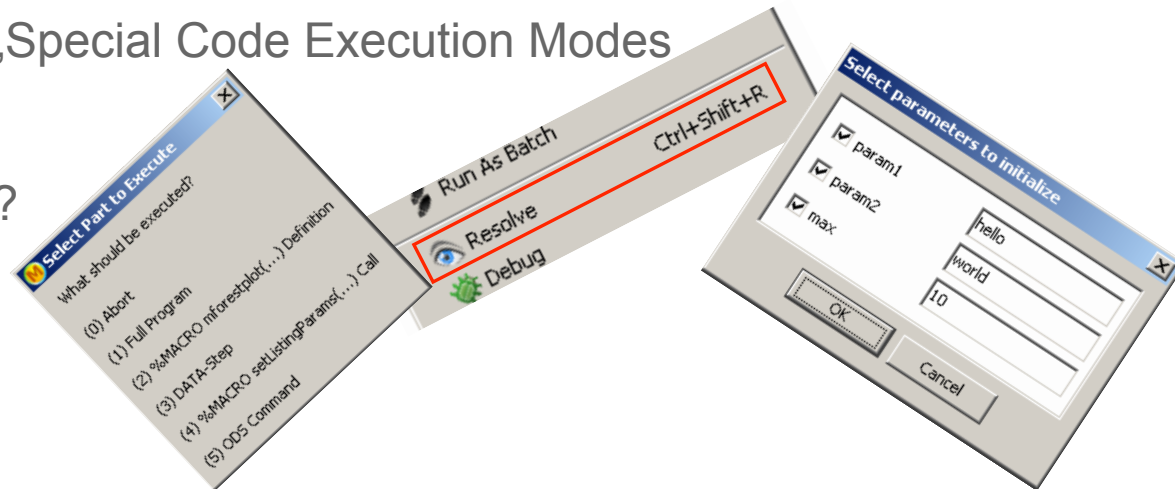
- Resolve Code
 - Can this be done automatically?
 - → Yes 😊



Macumba – Debugging made easy

- That's It For „Special Code Execution Modes

- Questions?



- Next is “DATA-Step Debugger”

```

DEBUGGER SOURCE
7 DATA class / DEBUG;
8   set sashelp.class;
9   DO i=1 to 2;
! 10     PUT i= name=;
11   END;
12 RUN;
    
```

```

1
2 DATA class;
3   SET sashelp.class;
4 DO i=1 to 2;
5     PUT i= name=;
6   END;
7 RUN;
8
    
```

Name: Name
 Value: Alfred
 Type: DATA

Macumba – DATA-Step Debugger – Overview - 1

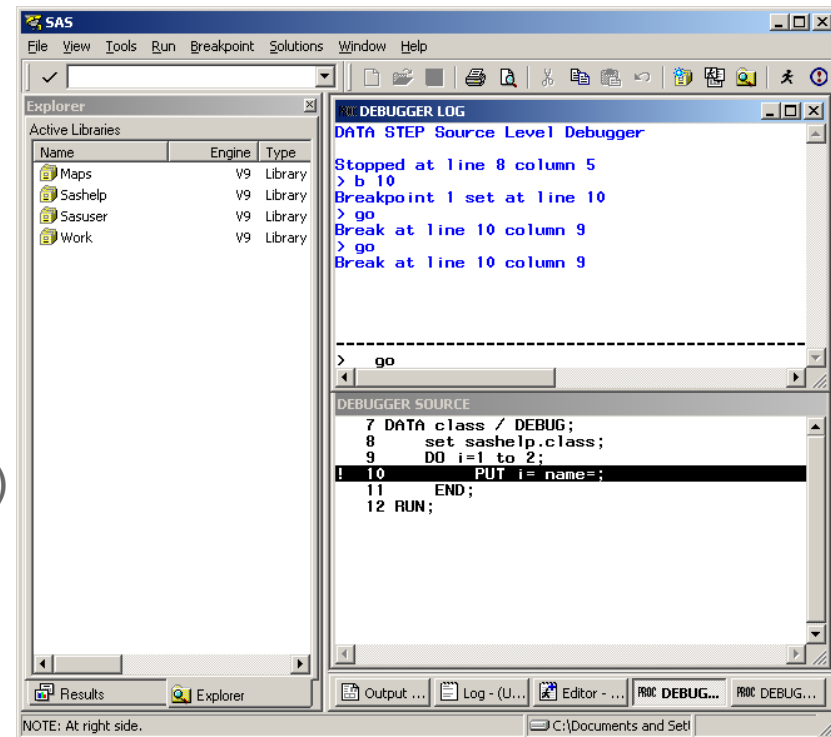


- Interactive SAS DATA-Step Debugger
 - Change Code (add / DEBUG parameter to DATA command)

SAS Code

```
DATA class / DEBUG;  
  SET sashelp.class;  
  DO i=1 to 2;  
    PUT i= name=;  
  END;  
RUN;
```

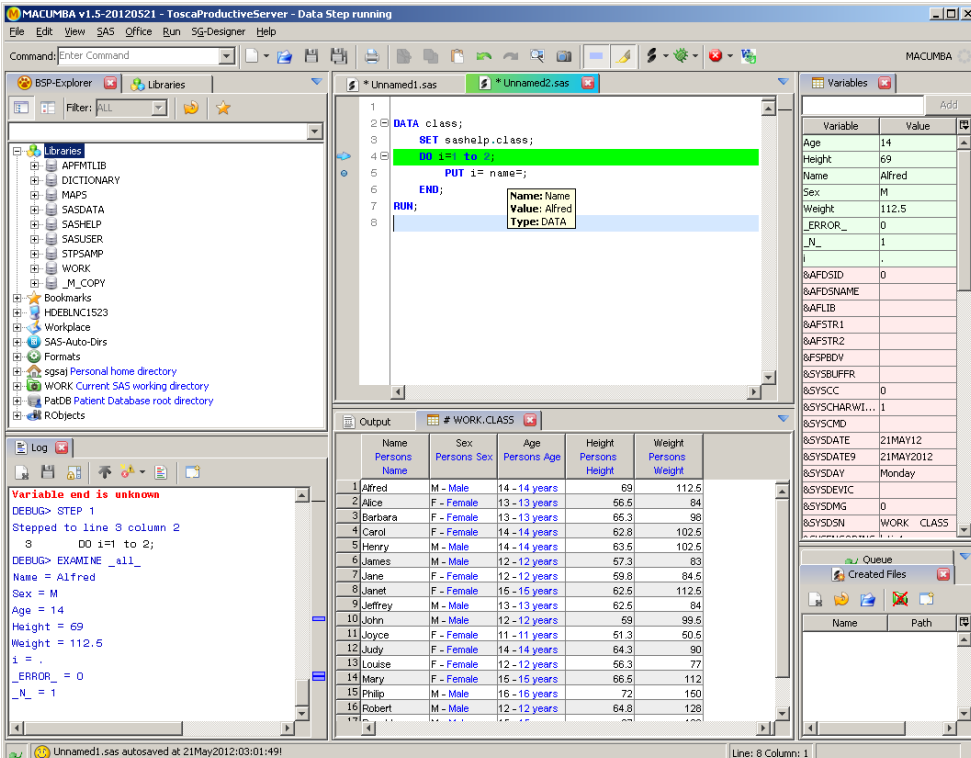
- Command Driven
 - STEP, GO, BREAK, EXAMINE
- Separate Window (No Syntax Highlight)



Macumba – DATA-Step Debugger – Overview - 2



- MACUMBA DATA-Step Debugger
 - Use  Button to start
 - Shortcut / Mouse based
 - Similar to Eclipse / VBA
 - In SAS Code Editor
 - Same Syntax Highlight



The screenshot displays the MACUMBA v1.5-20120521 - ToscaProductiveServer - Data Step running interface. The main window shows a SAS code editor with the following code:

```
1 DATA class;  
2 SET sashelp.class;  
3 DO i=1 TO 1;  
4 PUT i= name=;  
5 END;  
6 RUN;
```

The log window at the bottom left shows the following output:

```
Variable end is unknown  
DEBUG: STEP 1  
Stepped to line 3 column 2  
3 DO i=1 to 1;  
DEBUG: EXAMINE _all_  
Name = Alfred  
Sex = M  
Age = 14  
Height = 69  
Weight = 112.5  
i = .  
_ERROR_ = 0  
_N_ = 1
```

The variable explorer on the right shows the following variables and their values:

Variable	Value
Age	14
Height	69
Name	Alfred
Sex	M
Weight	112.5
ERROR	0
N	1
%AFDSID	0
%AFDSNAME	
%AFLIB	
%AFSTR1	
%AFSTR2	
%FSPEDV	
%SYSBUFFR	
%SYSCC	0
%SYSCHARW...	1
%SYSCMD	
%SYSDATE	21MAY12
%SYSDATE9	21MAY2012
%SYSDAY	Monday
%SYSDEVIC	
%SYSDMG	0
%SYSDSN	WORK CLASS

The data table at the bottom shows the following data:

Name	Sex	Age	Height	Weight
Alfred	M - Male	14 - 14 years	69	112.5
Alice	F - Female	13 - 13 years	56.5	84
Barbara	F - Female	13 - 13 years	65.3	98
Carol	F - Female	14 - 14 years	62.8	102.5
Henry	M - Male	14 - 14 years	63.5	102.5
James	M - Male	12 - 12 years	57.3	83
Jane	F - Female	12 - 12 years	59.8	84.5
Janet	F - Female	15 - 15 years	62.5	112.5
Jeffrey	M - Male	13 - 13 years	62.5	84
John	M - Male	12 - 12 years	59	99.5
Joyce	F - Female	11 - 11 years	51.3	50.5
Judy	F - Female	14 - 14 years	64.3	90
Louise	F - Female	12 - 12 years	56.3	77
Mary	F - Female	15 - 15 years	66.5	112
Philip	M - Male	16 - 16 years	72	160
Robert	M - Male	12 - 12 years	64.8	128

Macumba – DATA-Step Debugger – Implementation 1



- LDEBUG parameter added to DATA Command (internally)

```
SAS Code  
DATA class / LDEBUG;  
  SET sashelp.class;  
  DO i=1 to 2;  
    PUT i= name=;  
  END;  
RUN;
```

```
2 DATA class;  
3 SET sashelp.class;  
4 DO i=1 to 2;  
5 PUT i= name=;  
6 END;  
7 RUN;  
8
```

Name: Name
Value: Alfred
Type: DATA

- Auto EXAMINE variable for Tool Tip Text
- Execute related SAS command in Background
 - Move Debug line → JUMP
 - Breakpoint Add / Delete (DELETE) BREAKPOINT <line>
 - ...

Macumba – DATA-Step Debugger – Implementation 2



- Issues in Implementation

1. Q.: How to Find DATA-Step End?

A1.: Search for line with RUN;

SAS Code

```
DATA tmp;  
  a = step +  
    run;   
  PUT a=;  
RUN;
```

SAS Code

```
DATA tmp;  
  a = step + run;  
  PUT a=; RUN; 
```


Macumba – DATA-Step Debugger – Implementation 2



- Issues in Implementation

1. Q.: How to Find DATA-Step End?

A2.: <Semicolon> RUN <Semicolon> (ignore whitespaces)

SAS Code

```
DATA tmp;  
  SET sashelp.class;  
  /* simple copy of sashelp.class */  
RUN;
```

A3.: <Semicolon> RUN <Semicolon> (ignore whitespaces and comments)

SAS Code

```
DATA tmp;  
  %calc_macro(p1=abc, p2=def)  
RUN;
```

Macumba – DATA-Step Debugger – Implementation 2



- Issues in Implementation

1. Q.: How to Find DATA-Step End?
What about the following code?

SAS Code

```
DATA class;  
  SET sashelp.class;  
  DO i=1 to 2;  
    PUT i= name=;  
  END;
```

```
PROC SORT DATA=class;  
  BY sex age name;  
RUN;
```

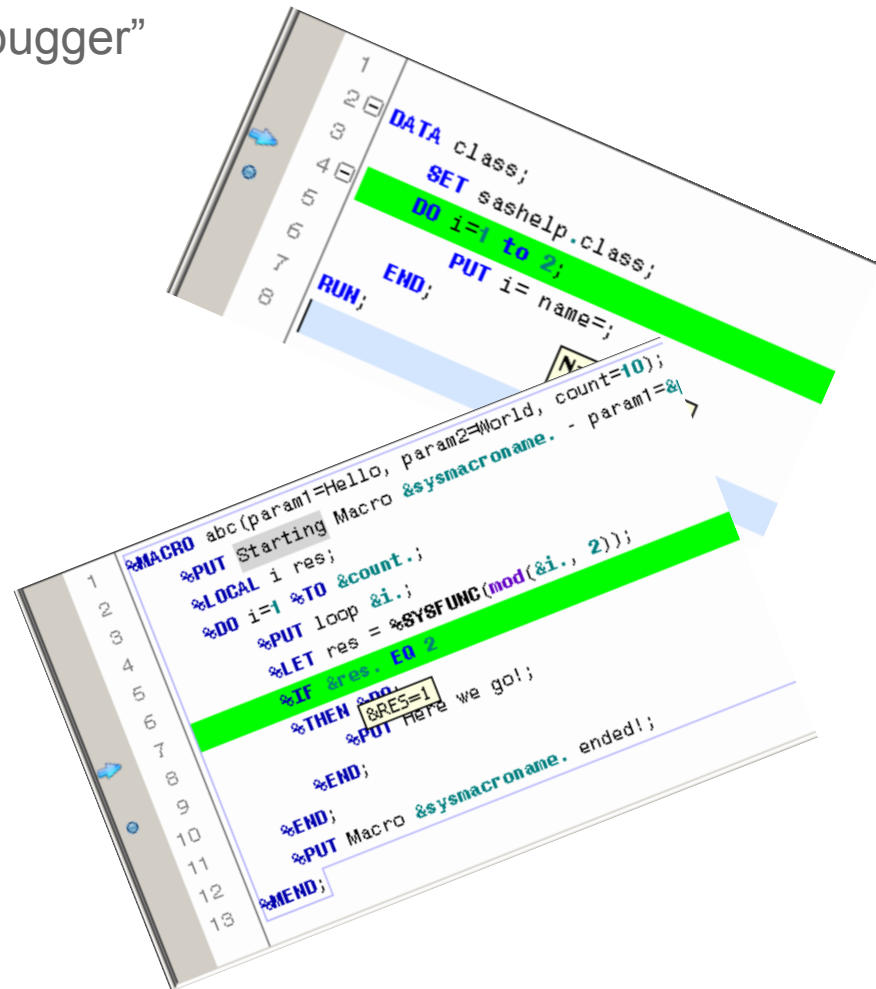
SAS Code

```
DATA _NULL_;  
  a=1;  
  %PUT %STR(%);RUN;);  
  PUT a=;  
RUN;
```

A4.: Parse complete code, create code groups → use DATA group

Macumba – Debugging made easy

- That's It For „DATA-Step Debugger”
- Questions?
- Next is “MACRO Debugger”



Macumba – MACRO Debugger – Overview 1



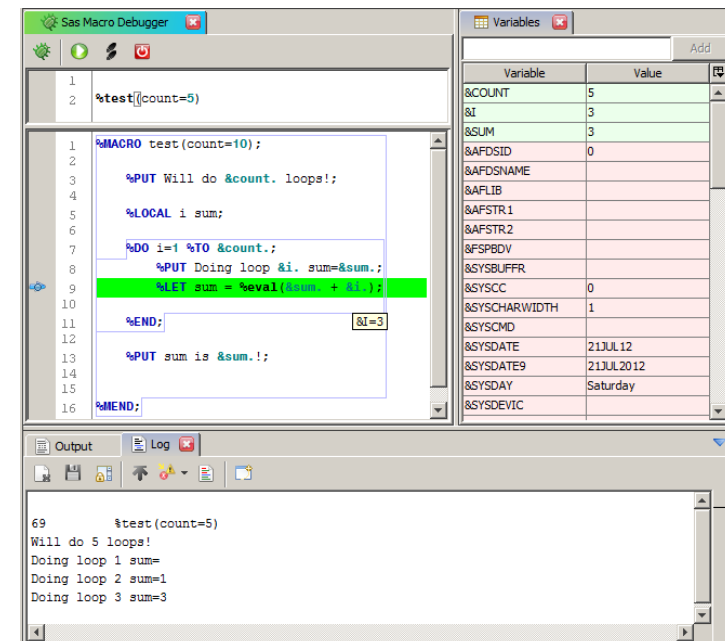
- How to debug a SAS macro normally?
 - Use extra debugging code (%PUT, %ABORT, ...)
 - Use SAS Options like SYMBOLGEN, MLOGIC, ...
 - Use something like “Trial and Error” or “Good Guessing”
- How to debug a SAS macro with MACUMBA?
 - Use the “SAS Macro Debugger” View

```
1 %MACRO abc(param1=Hello, param2=World, count=10);  
2   %PUT Starting Macro &sysmacroname. - param1=&1  
3   %LOCAL i res;  
4   %DO i=1 %TO &count.;  
5     %PUT loop &i.;  
6     %LET res = %SYSFUNC(mod(&i., 2));  
7     %IF &res. EQ 2  
8       %THEN DO  
9         %RES=1  
10        %PUT Here we go!;  
11      %END;  
12    %END;  
13  %PUT Macro &sysmacroname. ended!;  
14 %MEND;
```

Macumba – MACRO Debugger – Overview 2



- SAS Macro Debugger
 - Real interactive MACRO debugging
 - Step By Step Execution, Breakpoints, “Code Jumping”
 - Display and modification of local macro variable values
 - Execution of “external” SAS code within the macro
 - Debugging of multiple macros at the same time
 - e.g. one macro is calling another macro



Sas Macro Debugger

1
2 %test(count=5)

```
1 %MACRO test(count=10);  
2  
3   %PUT Will do &count. loops!;  
4  
5   %LOCAL i sum;  
6  
7   %DO i=1 %TO &count.;  
8     %PUT Doing loop &i. sum=&sum.;  
9     %LET sum = %eval(&sum. + &i.);  
10  
11   %END;  
12  
13   %PUT sum is &sum.!;  
14  
15  
16 %MEND;
```

Variables

Variable	Value
&COUNT	5
&I	3
&SUM	3
&AFDSID	0
&AFDSNAME	
&AFLIB	
&AFSTR1	
&AFSTR2	
&FSPBDV	
&SYSBUFFR	
&SYSCC	0
&SYSCHARWIDTH	1
&SYSCMD	
&SYSDATE	21JUL12
&SYSDATE9	21JUL2012
&SYSDAY	Saturday
&SYSDEVIC	

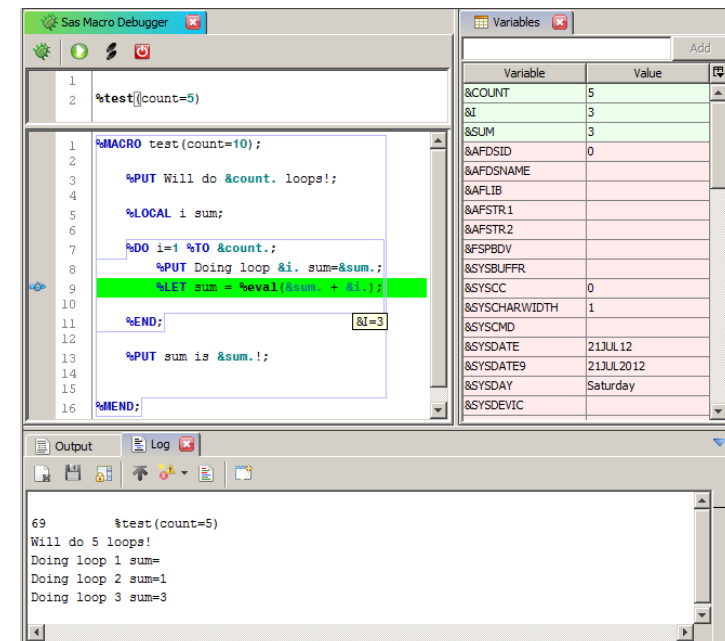
Output

```
69      %test(count=5)  
Will do 5 loops!  
Doing loop 1 sum=  
Doing loop 2 sum=1  
Doing loop 3 sum=3
```

Macumba – MACRO Debugger – Implementation 1



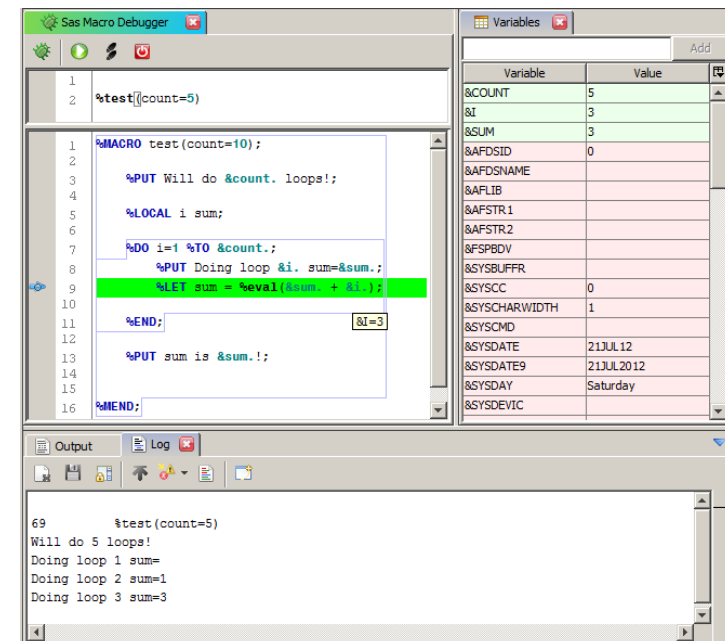
- Implemented as own View
 - Allows special invocations
 - E.g. from another macro or within a DATA-Step
- Similar usage as the DATA-Step Debugger
 - Start debugging by invoking
 - Double Click a line to set / remove a Breakpoint
 - Move the to jump in the code
 - Use or to continue code execution
 - Abort the macro execution by



Macumba – MACRO Debugger – Implementation 2



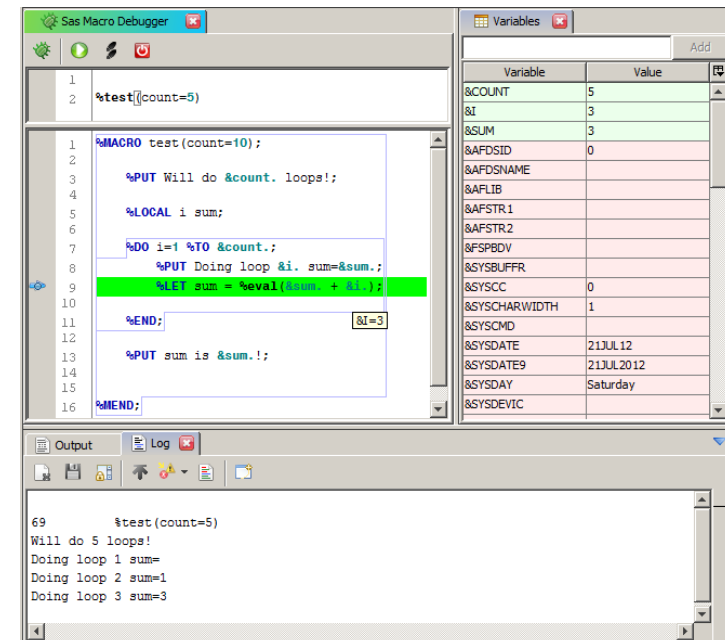
- Macro variables are resolved on “Mouse Over” and can be edited in the “Variables” view
- The log is updated simultaneously on code execution



Macumba – MACRO Debugger – Outlook



- Planned Features
 - Modification of macro code while debugging
 - Automatically add sub macros to debugger when invoked from debugging macro
 - Features available in the DATA-Step debugger (WATCH, BREAK ON, ...)
- Code Coverage Measurement
 - Search for code lines not invoked within validation process





Science For A Better Life

Thank you!