



Science For A Better Life

Macumba

State of the Art SAS GUI

Debugging made easy

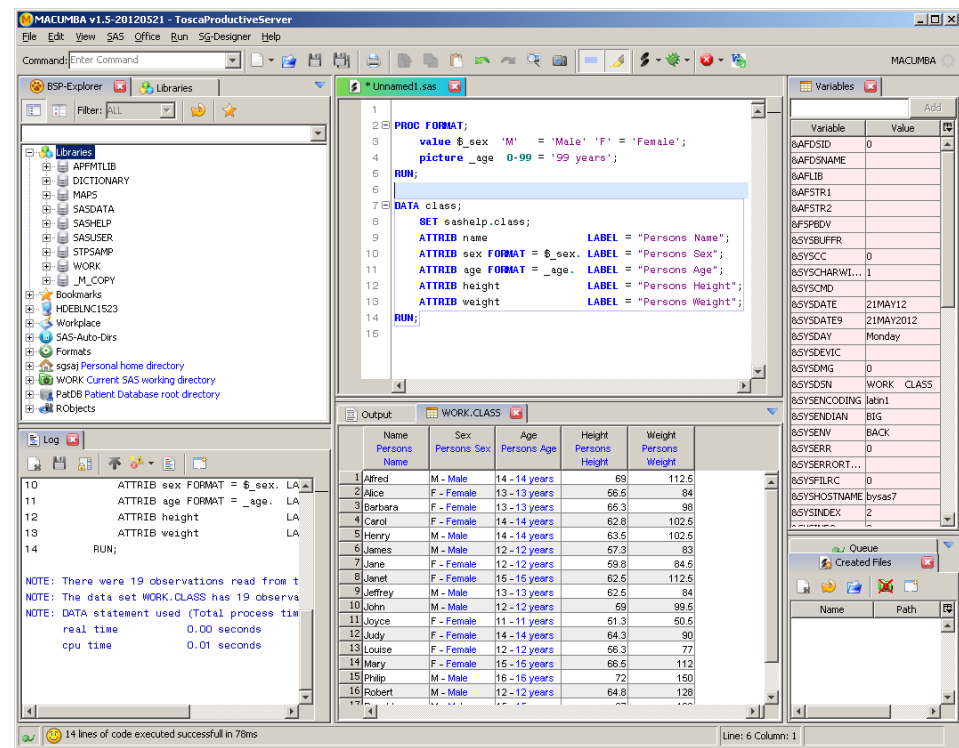
Macumba – Debugging made easy – Overview



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
 - Resolve Code
3. SAS DATA-Step Debugger
 - Overview
 - Compare with Interactive SAS Debugger
4. Outlook

Macumba – What is MACUMBA?

- In-House developed SAS IDE
- “Pseudo”-Interactive code execution on Remote SAS Server
- Focus on SAS program / macro development
- 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

Libraries: APFMTLIB, DICTIONARY, MAPS, SASDATA, SASHELP, SASUSER, STPSAMP, WORK, _M_COPY, HDEBLNC1523, Workplace, SAS-Auto-Dirs, Formats, sgsl Personal home directory, WORK Current SAS working directory, PatDB Patient Database root directory, Objects

```

1  PROC FORMAT;
2  value $ _sex 'M' = 'Male' 'F' = 'Female';
3  picture _age 0-99 = '99 years';
4  RUN;
5
6
7  DATA class;
8  SET sashelp.class;
9  ATTRIB name LABEL = "Persons Name";
10 ATTRIB sex FORMAT = $ _sex. LABEL = "Persons Sex";
11 ATTRIB age FORMAT = _age. LABEL = "Persons Age";
12 ATTRIB height LABEL = "Persons Height";
13 ATTRIB weight LABEL = "Persons Weight";
14 RUN;
15

```

Output: WORK.CLASS

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	112.6
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

Log:

```

10 ATTRIB sex FORMAT = $ _sex. LA
11 ATTRIB age FORMAT = _age. LA
12 ATTRIB height LA
13 ATTRIB weight LA
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

```

Variables: Variable, Value

Variable	Value
\$AFOSID	0
\$AFOSNAME	
\$AFLIB	
\$AFSTR1	
\$AFSTR2	
\$AFSTR3	
\$AFSTR4	
\$AFSTR5	
\$AFSTR6	
\$AFSTR7	
\$AFSTR8	
\$AFSTR9	
\$AFSTR10	
\$AFSTR11	
\$AFSTR12	
\$AFSTR13	
\$AFSTR14	
\$AFSTR15	
\$AFSTR16	
\$AFSTR17	
\$AFSTR18	
\$AFSTR19	
\$AFSTR20	
\$AFSTR21	
\$AFSTR22	
\$AFSTR23	
\$AFSTR24	
\$AFSTR25	
\$AFSTR26	
\$AFSTR27	
\$AFSTR28	
\$AFSTR29	
\$AFSTR30	
\$AFSTR31	
\$AFSTR32	
\$AFSTR33	
\$AFSTR34	
\$AFSTR35	
\$AFSTR36	
\$AFSTR37	
\$AFSTR38	
\$AFSTR39	
\$AFSTR40	
\$AFSTR41	
\$AFSTR42	
\$AFSTR43	
\$AFSTR44	
\$AFSTR45	
\$AFSTR46	
\$AFSTR47	
\$AFSTR48	
\$AFSTR49	
\$AFSTR50	
\$AFSTR51	
\$AFSTR52	
\$AFSTR53	
\$AFSTR54	
\$AFSTR55	
\$AFSTR56	
\$AFSTR57	
\$AFSTR58	
\$AFSTR59	
\$AFSTR60	
\$AFSTR61	
\$AFSTR62	
\$AFSTR63	
\$AFSTR64	
\$AFSTR65	
\$AFSTR66	
\$AFSTR67	
\$AFSTR68	
\$AFSTR69	
\$AFSTR70	
\$AFSTR71	
\$AFSTR72	
\$AFSTR73	
\$AFSTR74	
\$AFSTR75	
\$AFSTR76	
\$AFSTR77	
\$AFSTR78	
\$AFSTR79	
\$AFSTR80	
\$AFSTR81	
\$AFSTR82	
\$AFSTR83	
\$AFSTR84	
\$AFSTR85	
\$AFSTR86	
\$AFSTR87	
\$AFSTR88	
\$AFSTR89	
\$AFSTR90	
\$AFSTR91	
\$AFSTR92	
\$AFSTR93	
\$AFSTR94	
\$AFSTR95	
\$AFSTR96	
\$AFSTR97	
\$AFSTR98	
\$AFSTR99	
\$AFSTR100	

Queue: Created Files

Name Path

Line: 6 Column: 1

Macumba – Special SAS Code Execution Modes 1

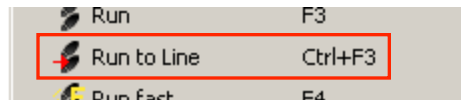


- PC-SAS provides two Execution Modes
 - Execute Selection (if code selected)
 - Execute All (if nothing selected)
- Uncomfortable when developing / debugging big programs
 - Execute from start to cursor position → scroll up / down
 - Execute from cursor position to program end → scroll down / up
 -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

Macumba – Special SAS Code Execution Modes 2



- Run To Line



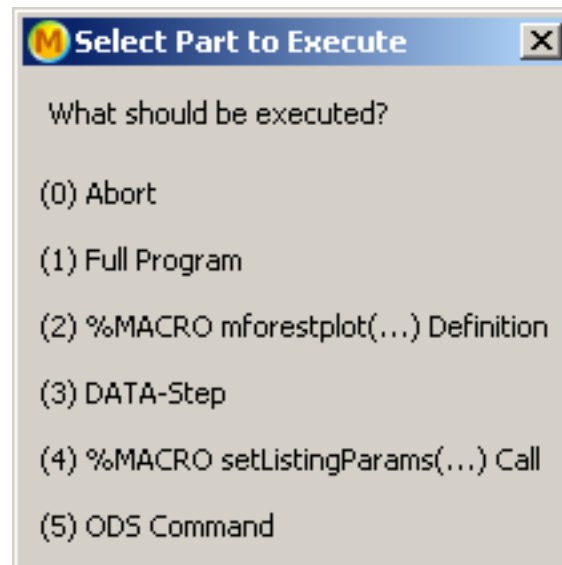
- Execute all code from start of program to cursor line

- Run From Line

- Execute all code from cursor line to end of program

- Run Step

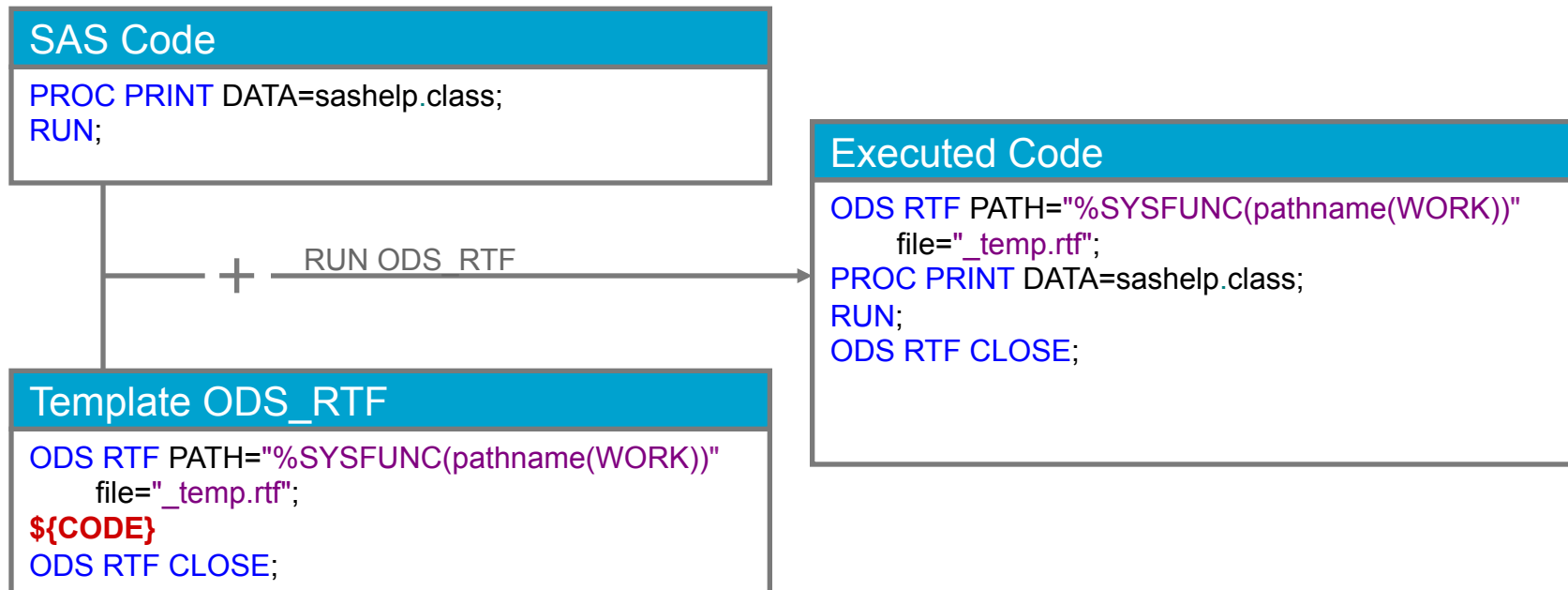
- Execute current step
 - %MACRO Definition
 - DATA- or PROC-Step
 - Global Command
 - Macro Call



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

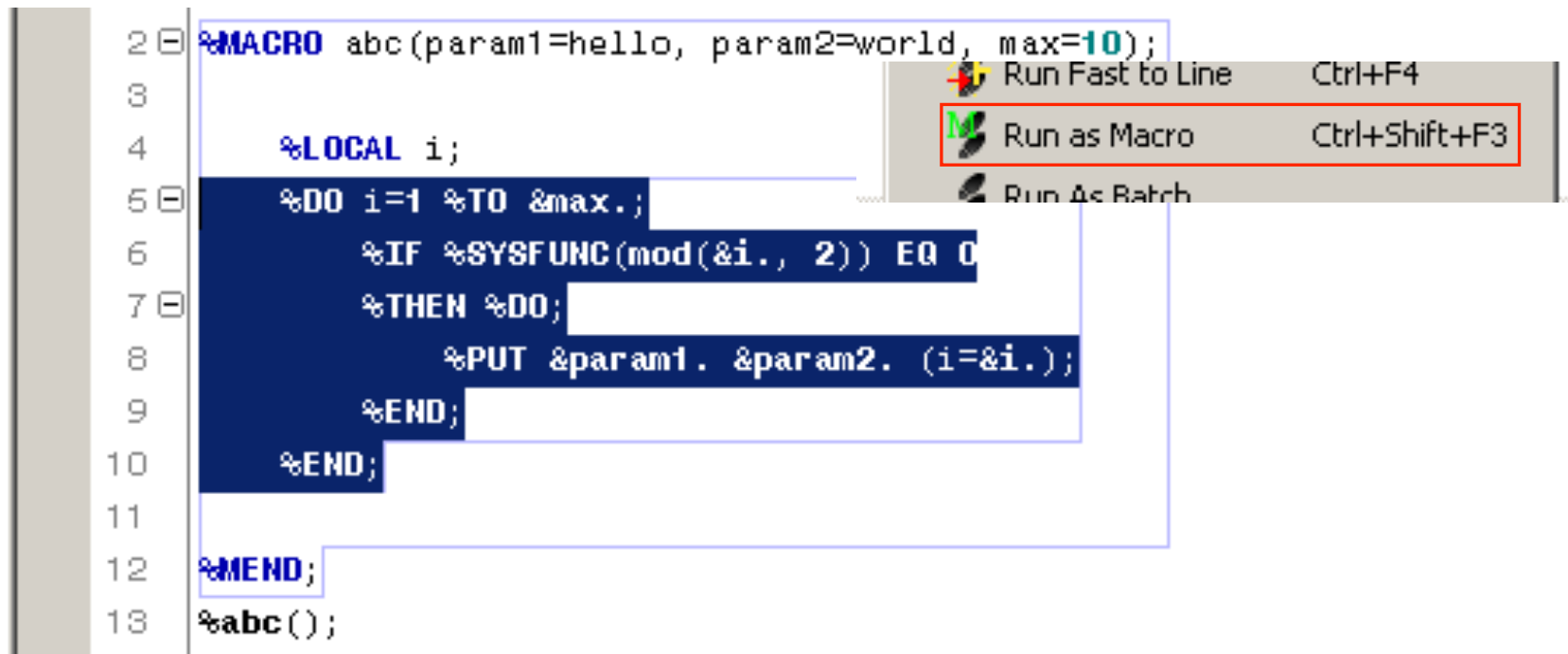
```
ERROR: The %DO statement is not valid in open  
code.  
ERROR: The %IF statement is not valid in open  
code.  
WARNING: Apparent symbolic reference PARAM1  
not resolved.  
WARNING: Apparent symbolic reference PARAM2  
not resolved.  
WARNING: Apparent symbolic reference I not  
resolved.
```

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 displays the SAS IDE interface. In the background, a SAS code editor shows a macro definition: `%MACRO abc(param1=hello, param2=world, max=10);` on line 2, followed by `%LOCAL i;` on line 4, and `%MEND;` on line 12. A dialog box titled "Select parameters to initialize" is open in the foreground. It contains three checked items: "param1" with the value "hello", "param2" with the value "world", and "max" with the value "10". Below the list are "OK" and "Cancel" buttons. To the right of the dialog, a window titled "SAS Code" displays the macro code generated by the "Run As Macro" action. The code is as follows:

```
%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

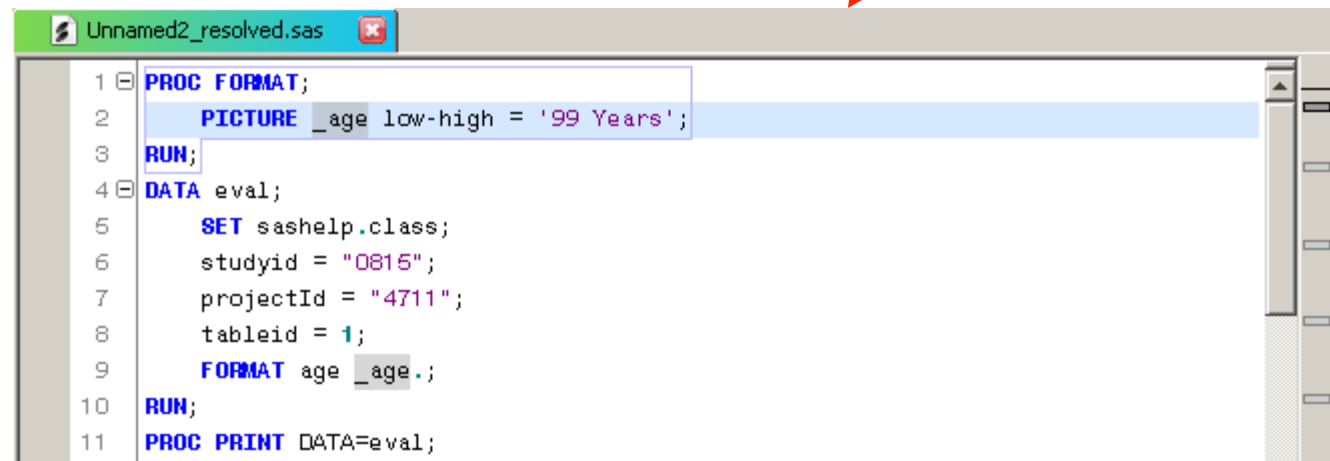
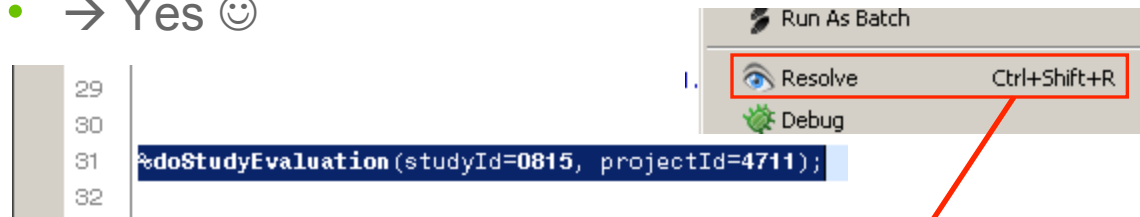
SAS Code (resolved.sas)

```
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



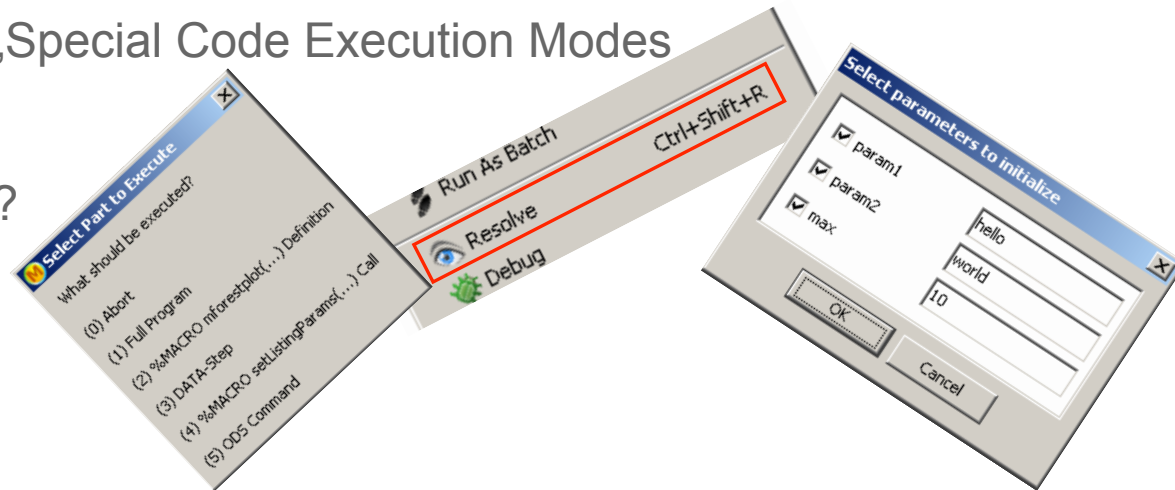
- Resolve Code
 - Can this be done automatically?
 - → Yes 😊



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

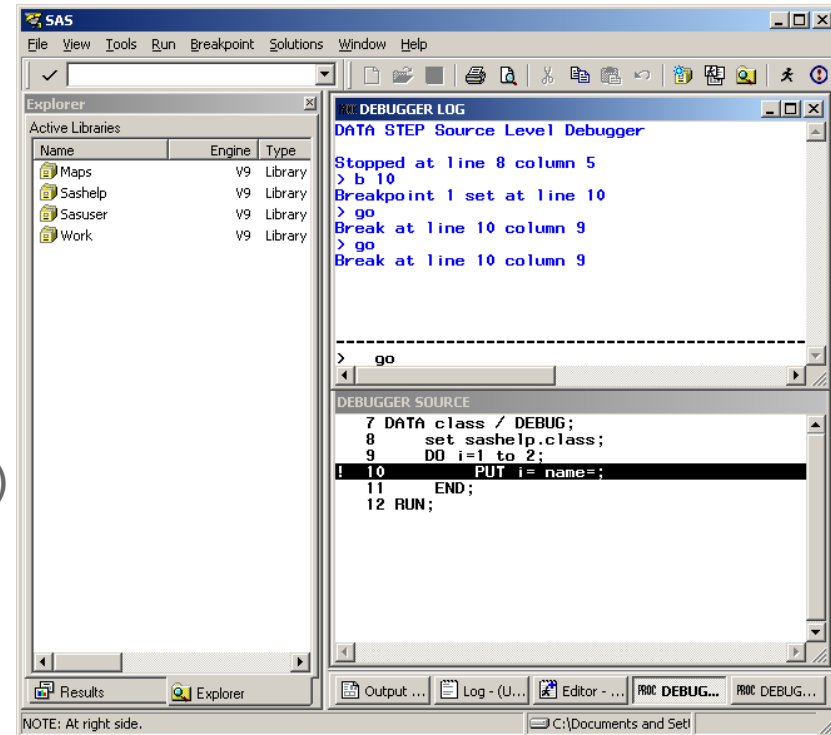


- Interactive SAS DATA-Step Debugger
 - Change Code (add / DEBUG parameter to DATA command)

SAS Code

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

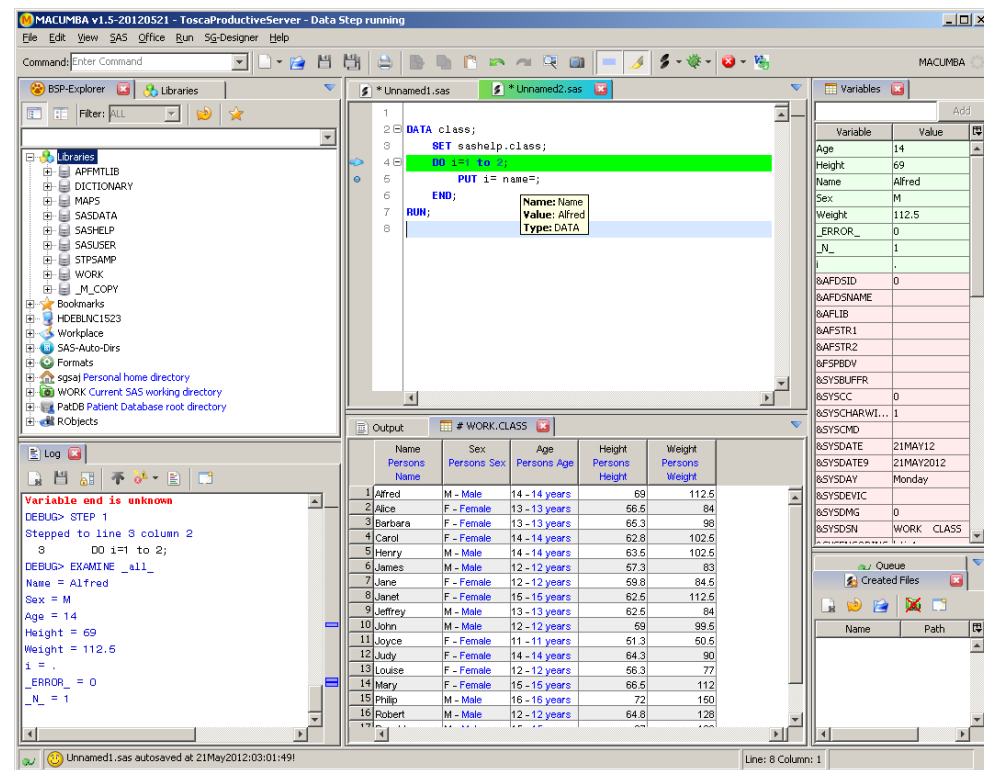
- Command Driven
 - STEP, GO, BREAK, EXAMINE
- Separate Window (No Syntax Highlight)



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



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 – DATA-Step Debugger – Implementation 2



- Issues in Implementation

- Q.: How to Determine if debugging ended?

A1.: Debugging ended after the user pressed the Quit Button

A2.: Scan log for “The DATA STEP program has completed execution”
→ issue QUIT command automatically afterwards

- Q.: How to debug the following code:

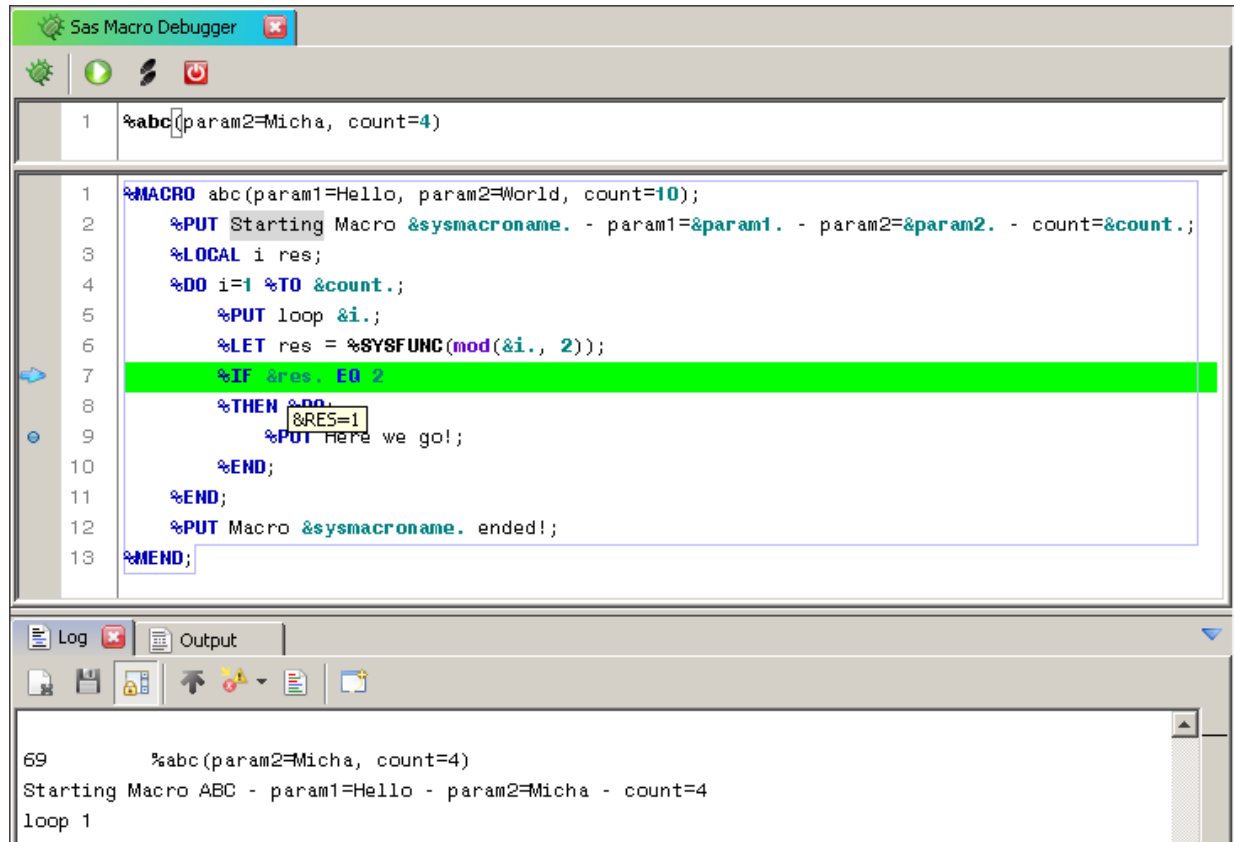
SAS Code

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

A.: Use Resolve Code before and debug resolved code 😊

Macumba – Outlook

- SAS Macro Debugger
 - Single Step debugging for SAS Macros (PhUSE 2012)



The screenshot shows the SAS Macro Debugger window. The top panel displays the macro code being debugged. The bottom panel shows the execution log.

```

1 %abc(param2=Micha, count=4)

1 %MACRO abc(param1=Hello, param2=World, count=10);
2   %PUT Starting Macro &sysmacroname. - param1=&param1. - param2=&param2. - count=&count.;
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         %PUT Here we go!;
10      %END;
11 %END;
12 %PUT Macro &sysmacroname. ended!;
13 %MEND;
  
```

The execution log shows the following output:

```

69      %abc(param2=Micha, count=4)
Starting Macro ABC - param1=Hello - param2=Micha - count=4
loop 1
  
```