

A Real Case Use of the Data Step Debugger

Diane Cunningham, Southern Company Services, Atlanta, GA

ABSTRACT

If your code has syntax errors, SAS® can be very useful in finding the problem. The SAS log produces many helpful notes to assist you. What if your code is technically sound, but is not producing the expected results? The Data Step Debugger is a nice interactive way to watch what's going on in the data step. This paper will describe how the Data Step Debugger helped to uncover the flaw in the program logic.

INTRODUCTION

At SESUG 2008, Paul Dorfman used the data step debugger as a means to teach HOWWW loops. This was the first time I had seen the debugger and thought it would be great to learn. It took several months, but I finally had code that ran without errors, but did not give the expected results. This is how I used the debugger to track down the flaw in my program logic.

BACKGROUND

I was provided with data collected hourly of the energy usage flowing through a sample of electric substations. I wanted to provide summary statistics on the data, including the percent of missing data and the number of zero use intervals. If the interval was missing, the data record did not exist. For example, if data for Jan 10th was not collected, this is how the data stream would appear:

Date	Hour	Value
. . .		
Jan 9	23	###
Jan 9	24	###
Jan 11	1	###
. . .		

My thought was that the easiest way to report on the missing intervals would be to first create records for the missing hours and fill the values with missing (null value). There are other ways I could have approached this problem, such as count the number of records between the date of the first record and last record and then compare that to the total number intervals that should have existed. Although this approach may have been less complicated, it would have limited my ability to report the date and hour of the first missing interval.

FIRST ATTEMPT AT SOLUTION

I used a do while loop to walk through the data, comparing the date and time of the current interval to the loop counter. If they were in-sync, the data existed. If the date and time of the current interval was out-of-sync with the counter, there were missing intervals. Here is the original code.

```
data two (drop=cal_dte dttm rdng);
  retain cal_dte dst;
  set here.test_1810;
  by preempt mtrpt;
  if first.mtrpt then
    cal_dte = datetime; ** set the loop counter to the datetime of first record
  **;
  if datetime ne cal_dte then do;
    ** If interval is missing, set values of current record to temporary
    variables. Loop through all the missing intervals setting the value to missing
    **;
    dttm = datetime;
    rdng = reading;
    do until (dttm = cal_dte);
      datetime = cal_dte;
      reading = .;
    end;
  end;
```

```

        output;
        cal_dte = intnx('hour',cal_dte,1);
    end;
    datetime = dtm; * reset the values from the temporary variables, and output
*;
    reading = rdng;
    output;
    cal_dte = intnx('hour',cal_dte,1);
end;
else do; ** if interval not missing, output and increment the loop counter **;
    output;
    cal_dte = intnx('hour',cal_dte,1);
end;
if last.mtrpt and datetime ne dhms('31dec08'd,23,0,0) then do;
** If the last record is not the last hour of the year, fill with missing **;
do until (cal_dte = dhms('31dec08'd,23,0,0));
    datetime = cal_dte;
    reading = .;
    output;
    cal_dte = intnx('hour',cal_dte,1);
end;
end;

run;

```

USING THE DEBUGGER

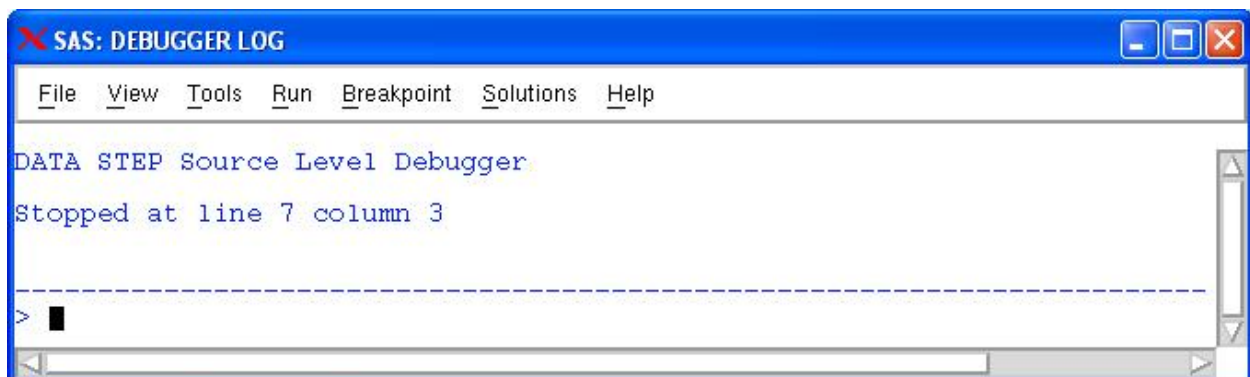
Can you see the problem? It wasn't obvious to me either, but the program was stuck in the do until loop. I ran this code on a UNIX system in batch mode, but the program would not finish. I was forced to kill the process, which left me with an empty log. I looked over the program again and again, but could not explain what caused the infinite loop. I had to confirm the logic was correct. The Data Step Debugger allows the user to validate the program logic in an easy interactive way. The Debugger allows you to step through the data step and examine values assigned to variables. The Data Step Debugger is documented in the Appendix of the SAS 9.2 Language Reference: Dictionary.

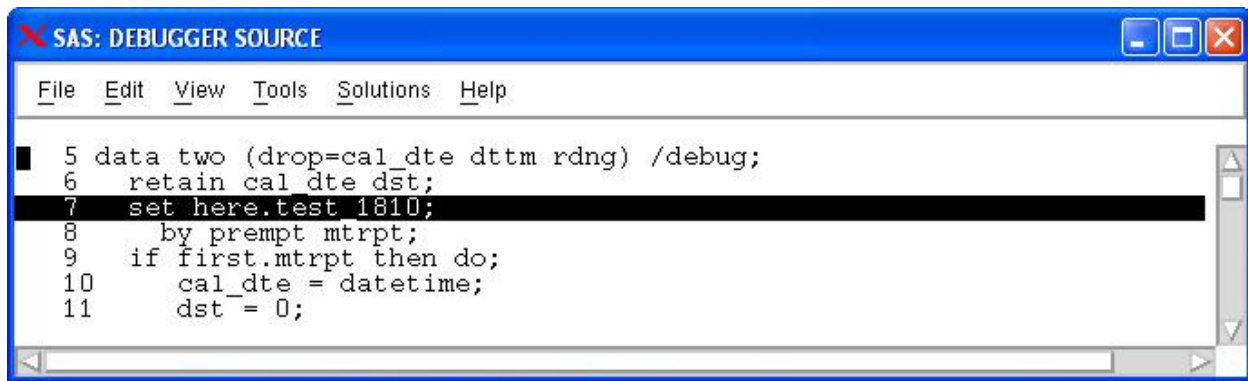
STARTING THE DEBUGGER

All you need to do to invoke the debugger is add a parameter to the DATA statement:

```
data two (drop=cal_dte dtm rdng) / debug;
```

When you submit the data step, an interactive window is displayed that allows the user to control the action of the data step. In the UNIX system, these windows look like this:





The Source window displays SAS code and highlights the current line being executed. The log window shows the results from executing the code. The command line is at the bottom of the log window and is used to issue commands to the debugger. As you enter commands, you can watch the highlighted row change in the source window and read the results of the action in the log window.

BASIC DEBUGGER COMMANDS

The following is a description of just some of the commands available with the debugger. They are the most basic and were the ones used to validate my code. Again, a full description of the debugger commands can be found in the SAS Language Reference: Dictionary.

STEP (OR ST)

Executes one statement at a time, or add a number to execute that number of statements. By default, the ENTER key is programmed to execute STEP 1.

GO (OR G)

Starts or resumes execution of the DATA step. Execution continues until a breakpoint is encountered, until the value of a watched variable changes, or until the DATA step completes.

EXAMINE (OR EX)

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.

WATCH (OR W)

Suspends execution when the value of a variable changes. The log displays the old value, new value, and the line number.

DELETE (OR D)

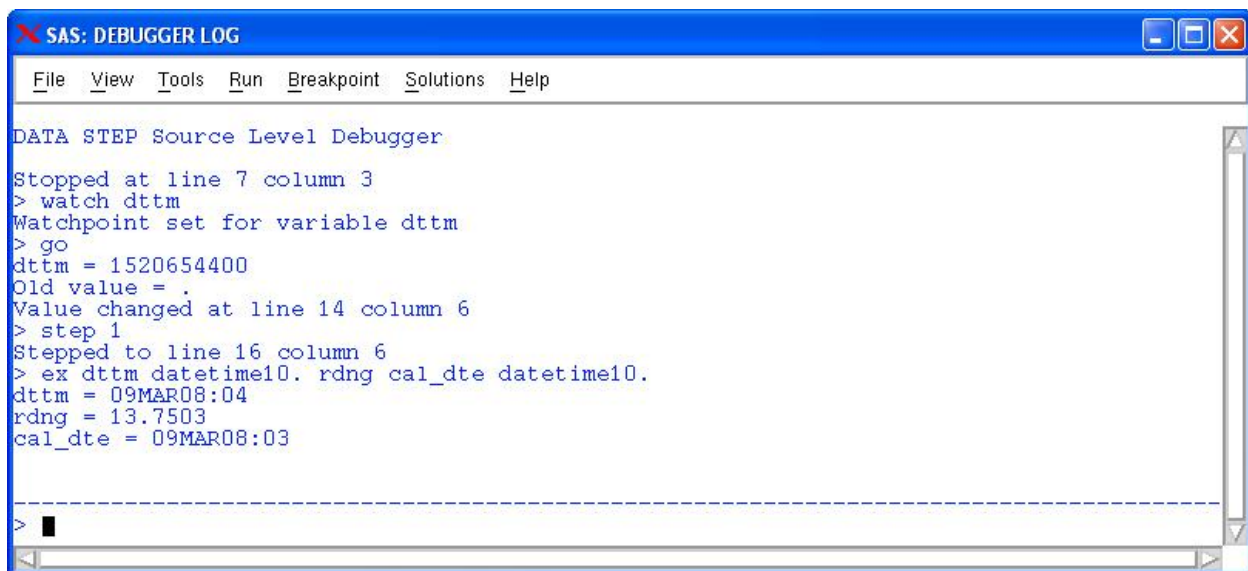
Deletes breakpoints or the watch status of variables in the DATA step.

QUIT (OR Q)

Terminates a debugger session and returns control to the SAS session.

EXAMPLE OF ISSUING DEBUGGING COMMANDS

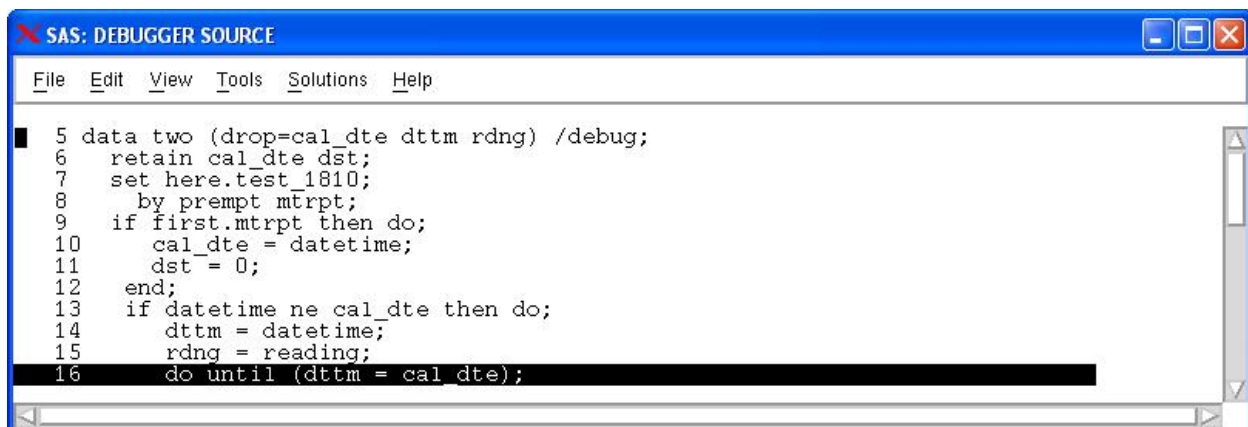
In the above code, the dttm variable is set only when the datetime of the current record does not match the loop counter variable cal_dte. I wanted to step through the logic of the code when a missing interval was found. I set a breakpoint when the value of dttm changes using the watch command. The following shows how the debugger log window displays the results of the watch, go, step and examine commands:



```
SAS: DEBUGGER LOG
File View Tools Run Breakpoint Solutions Help

DATA STEP Source Level Debugger
Stopped at line 7 column 3
> watch dttm
Watchpoint set for variable dttm
> go
dttm = 1520654400
Old value = .
Value changed at line 14 column 6
> step 1
Stepped to line 16 column 6
> ex dttm datetime10. rdng cal_dte datetime10.
dttm = 09MAR08:04
rdng = 13.7503
cal_dte = 09MAR08:03
>
```

The commands typed at the command line are echoed in the debugger log at the lines with the right arrow >. The log window shows us that the first missing period starts March 9, 2008 at 3 am and lasts one hour, which happens to be the start of daylight savings time. The debugger source window follows and shows current line of code after these commands were issued:



```
SAS: DEBUGGER SOURCE
File Edit View Tools Solutions Help

5 data two (drop=cal_dte dttm rdng) /debug;
6   retain cal_dte dst;
7   set here.test_1810;
8   by preempt mtrpt;
9   if first.mtrpt then do;
10    cal_dte = datetime;
11    dst = 0;
12  end;
13  if datetime ne cal_dte then do;
14    dttm = datetime;
15    rdng = reading;
16    do until (dttm = cal_dte);
```

I continued using the step and examine commands to follow the code within the do until loop and found that the program created a record for the missing interval exactly as designed. I also created a test dataset that ended prior to December 31st in order to confirm the second data filling code section worked correctly. The Debugger also verified that section of the code.

SOLUTION

So I still did not have a solution to my problem. Using the debugger only confirmed that the program correctly filled in the missing intervals, but left to run on its own it still got stuck in an infinite loop. Fortunately, I used the debugger in interactive mode. In this mode, even after I had to cancel the submitted statements to terminate the loop, SAS still provided the log. The log showed how many records from the source dataset were read and how many records were written to the output dataset before the statements were canceled. I mentioned earlier that I had been running this code in batch mode. The only way I knew how to stop the infinite loop was to kill the process and that left an empty log. I never was able to see the following log window after canceling the submitted statements:

```

SAS: Log-Untitled
File Edit View Tools Solutions Help

run;

NOTE: The data step has been abnormally terminated.
NOTE: There were 7085 observations read from the data set HERE.TEST_1810.
WARNING: The data set WORK.TWO may be incomplete.  When this step was stopped there
        were 335834 observations and 35 variables.
NOTE: Compressing data set WORK.TWO decreased size by 56.82 percent.
        Compressed is 3086 pages; un-compressed would require 7146 pages.
NOTE: DATA statement used (Total process time):
      real time           17.76 seconds
      cpu time            3.03 seconds

```

You can see that in a very short time (17.76 seconds) the program outputted over 335,000 records. The test dataset contained one substation. A full set of hourly data for one year for 2008 (a leap year) is 8,784 records. You may also notice that only 7,085 records were read from the input data set TEST_1810. Why did it stop there, what is special about that record? This record is Nov. 2, 2008 at 2 am. Do you know what is special about that day and hour? It is the end of daylight savings time. The clocks are set back one hour at 2 am. In recording hourly energy consumption, we end up with two records with the same time stamp. When the loop counter incremented to 3 am, the source data stayed at 2 am., falling behind the loop counter. The do until loop condition:

```
do until (dttm = cal_dte);
```

can never be met because the source data (dttm) stays at 2 am and the loop counter (cal_dte) continues to increment. Aha, problem solved, before incrementing the loop counter, first check if we are at the end of daylight savings time. If it's the first occurrence, identified by a new flag variable DST=0, do not increment the loop counter and set DST = 1. If it's not the end of daylight savings time, or DST = 1, then increment the loop counter.

FINAL PROGRAM

```

data two (drop=cal_dte dttm rdng);
  retain cal_dte dst;
  set here.test_1810;
  by premt mtrpt;
  if first.mtrpt then do;
    cal_dte = datetime; ** set the loop counter to the datetime of first record
  **;
  dst = 0;
end;
if datetime ne cal_dte then do;
  dttm = datetime;
  rdng = reading;
  do until (dttm = cal_dte);
    datetime = cal_dte;
    reading = .;
    output;
    if cal_dte = dhms('02nov08'd,2,0,0) and dst = 0 then do;
      dst = 1;
    end;
    else cal_dte = intnx('hour',cal_dte,1);
  end;
  datetime = dttm;
  reading = rdng;
  output;
  if cal_dte = dhms('02nov08'd,2,0,0) and dst = 0 then do;
    dst = 1;
  end;
end;

```

```

        end;
        else cal_dte = intnx('hour',cal_dte,1);
    end;
else do;
    output;
    if cal_dte = dhms('02nov08'd,2,0,0) and dst = 0 then do;
        dst = 1;
    end;
    else cal_dte = intnx('hour',cal_dte,1);
end;
if last.mtrpt and datetime ne dhms('31dec08'd,23,0,0) then do;
    do until (cal_dte = dhms('31dec08'd,23,0,0));
        datetime = cal_dte;
        reading = .;
        output;
        if cal_dte = dhms('02nov08'd,2,0,0) and dst = 0 then do;
            dst = 1;
        end;
        else cal_dte = intnx('hour',cal_dte,1);
    end;
end;
run;

```

REFERENCES

1. S. David Riva, "How to Use the Data Step Debugger." Proceedings of the Twenty-Fifth Annual SAS Users Group International Conference, Paper 52-25, Indianapolis, IN, USA 2000.
2. SAS Institute, Inc., SAS 9.2 Language Reference: Dictionary, 2009

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Diane Cunningham
 Southern Company Services
 241 Ralph McGill Blvd.
 Bin 10206
 Atlanta, GA 30308
 Work Phone: 404-506-3220
 Fax: 404-506-3180
 E-mail: dmcunnin@southernco.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
 Other brand and product names are trademarks of their respective companies.