

Resetting Variables to Zero

Ed Heaton, Data & Analytic Solutions

Abstract

SAS® provides a nice way to set a collection of variables to missing, and that's very handy. However, we often want to reset accumulator variables to zero when we are doing **By** group processing.

Traditionally, programmers code a **Do** loop to cycle through the variables that they want to reset -- assigning zero to each variable along the way. If we are processing hundreds of millions of records with millions of **By** groups, even resetting as few as five accumulator variables can be quite a load on our system.

The author proposes a macro – **%callZero()** – that will reset all of the accumulator variables to zero in one operation without loop processing.

Keywords: **addr()**, **peekC()**, **poke()**.

The Problem

Suppose we are performing **By** group processing to summarize data. This is a common task, many of do this sort of thing every day. We initialize an accumulator variable to zero at **first.byVariable** and then accumulate until **last.byVariable**, when we output.

```
Data New ;
  Set Foo ;
  By Key ;
  Retain X7-X13 0 ;
  Array x[*] X7-X13 ;
  If first.Key then do i=1 to dim(x) ;
    Drop i ;
    x[i] = 0 ;
  End ;
  X7 + X1 ;
  X8 + X2 ;
  X9 + X3 ;
  X10 + X4 ;
  X11 + X5 ;
  X12 + X6 ;
  X13 + 1 ;
  If last.Key then output ;
Run ;
```

However, consider the situation where we are resetting multiple accumulator variables to zero. We need either a group of assignment statements or an array and a loop to set the variables to zero. Suppose we

are working with very large datasets, say 500 million or more rows. Furthermore, suppose the **By** group ends with an identifier that changes often – say a person identifier. Then we will have to execute the loop to reset the accumulator variables very often.

A test on my Windows Vista machine running SAS 9.2 took almost six minutes to set five variables to zero a total of 1,000,000,000 times.

The Solution

Wouldn't it be nice if we could set them all to zero en masse? SAS provides a **call** routine called **call missing()** that enables us to set a collection of variables to missing en masse. But they provided no such **call** routine for setting them to zero.

At the last SAS Global Forum, Paul Dorfman talked about the **peekC()** function and **call poke()**. We can use these. A few details that make this possible.

- The **call poke()** routine writes to a block of memory when we specify the length of the block and the starting location. It writes bits. It doesn't know if it's writing character variable or numeric values and it doesn't care.
- All numeric variables are stored in eight bytes of memory. Always.
- SAS stores retained variables in contiguous memory. Our accumulator variables must be retained for them to accumulate across rows of our input dataset. So, if we declare all of our accumulator variables with a single **Retain** statement they will occupy contiguous memory. E.g.:

```
Retain a1-a5 0 ;
```

- The **peekC()** function reads from contiguous memory and returns the value without any translation. It doesn't really care if the value it's reading is supposed to represent a character string, a number, or a collection of numbers. It just returns the value.
- A **_temporary_** array stores its values in contiguous memory and it also retains the values across iterations of the **Data** step.

So, we can use the **Retain** statement to group our accumulator variables in memory, a **_temporary_** array to store a collection of zeros in contiguous memory, the **peekC()** function to pull the zeros from the array location, and **call poke()** to drop them into the accumulator variables. The only other thing we need is the **addr()** function to find the location in memory for the array of zeros and the accumulator variables.

```

Data New ;
  Set Foo ;
  By Key ;
  Retain X7-X13 0 ;
  If first.Key then do ;
    Array zero[7] _temporary_ ( 7*0 ) ;
    Call poke( peekC( addr(zero[1]) , 7*8 ) , addr(X7) ) ;
  End ;
  X7 + X1 ;
  X8 + X2 ;
  X9 + X3 ;
  X10 + X4 ;
  X11 + X5 ;
  X12 + X6 ;
  X13 + 1 ;
  If last.Key then output ;
Run ;

```

Unfortunately, this method was slower than using the loop. It took almost nine minutes to set the five numeric variables to zero the same 1,000,000,000 times.

So, our program would set our accumulator variables to zero with the following.

```
Data New ;
  Set Foo ;
  By Key ;
  Retain X7-X13 0 ;
  If first.Key then %callZero( firstVar=X7 , n=7 ) ;
  X7 + X1 ;
  X8 + X2 ;
  X9 + X3 ;
  X10 + X4 ;
  X11 + X5 ;
  X12 + X6 ;
  X13 + 1 ;
  If last.Key then output ;
Run ;
```

A Much Faster Way

There is another solution, however, that was discovered while working on this paper.

We are accumulating our data using the sum statement. The sum statement implicitly uses the **sum()** function. This function sums all the nonmissing values. So, if we reset our values to missing rather than to zero, our results will be the same unless we only add missing values throughout the entire **By** group. This might not be too onerous. If we can live with this, we can use the **call missing()** routine to reset our accumulator variables.

Call missing() is fast. Very fast. SAS can set five numeric variables to missing a total of 1,000,000,000 times with **call missing()** in less than a minute. And the results are identical unless all of the values you accumulate are missing.

```
Data New ;
  Set Foo ;
  By Key ;
  Retain X7-X13 0 ;
  If first.Key then call missing( of X7-X13 ) ;
  X7 + X1 ;
  X8 + X2 ;
  X9 + X3 ;
  X10 + X4 ;
  X11 + X5 ;
  X12 + X6 ;
  X13 + 1 ;
  If last.Key then output ;
Run ;
```

Conclusion

The call poke() routine can write to multiple variables in memory faster than looping through variables and writing to them one at a time. Call poke() doesn't know or care about the data type of the variables in memory; it just writes the bits. We can use this method to initialize variables to zero or special missing values or character strings.

However, we should consider whether it will work for us to initialize the values to missing. That's the fastest method we've found.

Contact Information

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

Ed Heaton
Data and Analytic Solutions, Inc.
3057 Nutley Street
Fairfax, VA 22031
Voice: 301-520-7414
Email: EHeaton@DASConsultants.com

The content of this paper is the work of the author and does not necessarily represent the opinions, recommendations, or practices of Data and Analytic Solutions, Inc. 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.

Appendix: Performance Testing

```

Options fullSTimer ;
%let max = 1e9 ;

Data _null_ ;
  Retain x1-x5 0 ;
  Array x[5] ;
  Do i=1 to &max ;
    Do j=1 to dim(x) ;
      x[j] = 0 ;
    End ;
  End ;
Run ;

Data _null_ ;
  Retain x1-x5 0 ;
  Array x[5] ;
  Array __[5] _temporary_ ( 5*0 ) ;
  Do i=1 to &max ;
    Call poke( peekC( addr(__[1]) , 5 * 8 ) , addr(x1) ) ;
  End ;
Run ;

/*=====*/
%macro callZero( firstVar= , n=1 ) ;
  call poke( "%sysFunc( repeat( 0 , &n*16-1 ) )"x , addr(&firstVar) )
%mEnd callZero ;
/*=====*/

Data _null_ ;
  Retain x1-x5 0 ;
  Do i=1 to &max ;
    %callZero( firstVar=x1 , n=5 ) ;
  End ;
Run ;

Data _null_ ;
  Retain x1-x5 0 ;
  Do i=1 to &max ;
    Call missing( of x1-x5 ) ;
  End ;
Run ;

```