ABSTRACT
This paper describes an application of the SAS hash object that captures and writes to a data set the values of SAS automatic variables and other variables created with statement options that are not normally output. While this is an academic exercise it illustrates some useful hash techniques including dynamic declaration of hash object data elements using CALL VNEXT.

KEYWORDS
HASH, VNEXT, “FIRST.variable”, “LAST.variable”.

INTRODUCTION
One day on SAS-L someone asked: Is there an option to tell SAS to output to a data set, automatic variables, that it normally would not output? Examples of such variables are _N_, _ERROR_, and any FIRST.variables and LAST.variables, plus many others created with the FILE and INFILE statements or as options on SET, MERGE and UPDATE statements. Of course there is no such option, but several suggestions were offered to assign the values to user-defined variables and then output them normally. This a fine solution but what if a more automatic solution is desired: something where all variables are output with no user-defined assignments?

One suggestion was to create a HASH object and define all variables found in the program data vector as data elements in the hash. Then instead of outputting the observations load them into the hash, using an index variable, such as _N_ that will keep each record unique. In this application the hash is essentially a dynamic array, where all elements of the array are declared at run time. While the code is more complex, it has the advantage of being dynamic. A hash is created using all variables found in the program data vector including those not otherwise output. This is possible because key and data elements for the hash are declared when the data step executes. The call routine VNEXT is used to retrieve the names of the variables found in the program data vector.

EXAMPLE
Consider a simple example with just a few, otherwise not output variables, _N_, _ERROR_, FIRST.variables, LAST.variables, INCLASS, END and NOBS.

```sas
set sashelp.class(in=inClass) end=end nobs=nobs;
by sex notsorted name age;
bmi = 703 * weight / height**2;
```

This bit of code will read all the data from SASHELP.CLASS and the associated by statement will create FIRST.variables and LAST.variables. The SET statement also creates three more variables: INCLASS, END, and NOBS. END will be used to detect the end-of-file so that the data in the hash object can be output to a SAS data set and the program stopped. INCLASS and NOBS are included just for the purpose of the example.

If we take this code snippet and embed it in other code that will declare the hash, update it for every observation and output the data results at the end we have a dynamic solution to output variables that are not otherwise output.
options validvarname=any;

data _null_;  

if _n_ eq 1 then do;  
attrib _PDVNAME_ length=64;  
declare hash _PDVHASH_(ordered:'Y');  
do while(1);  
call vnext(_PDVNAME_);  
putlog _PDVNAME_=;  
if missing(_PDVNAME_) then leave;  
if _PDVNAME_ eq '_'PDVNAME_'' then continue;  
if _PDVNAME_ eq '_N'_  
   then _PDVHASH_.defineKey(_PDVNAME_);  
   _PDVHASH_.defineData(_PDVNAME_);  
end;  
_PDVHASH_.defineDone();  
end;

set sashelp.class(in=inClass) end=end nobs=nobs;  
by sex notsorted name age;  
bmi = 703 * weight / height**2;

_PDVHASH_.add();  
if end then do;  
   _PDVHASH_.output(dataset:'work.AllVarsInPDV');  
stop;  
end;
run;

1. Set the SAS system option validvarname=ANY so the program can create variable names like "FIRST.SEX", which are not normally valid variable names.
2. Use null data step, output is done with the OUTPUT method.
3. Execute the following statements just one time on first iteration of the data step.
4. Declare a variable to receive the variable names returned by CALL VNEXT.
5. Declare the hash object and name it _PDVHASH_. Ordered:'Y' keeps the hash ordered by the ascending values of the KEY.
6. Start an infinite loop. Exit is provided for in line 9.
7. Get a variable name. Each time CALL VNEXT is executed it returns the "next" variable name from the program data vector.
8. Write it on the LOG.
9. End the loop if the value returned by CALL VNEXT is missing. When CALL VNEXT returns MISSING all variables have been processed.
10. Don't include _PDVNAME_ in the hash object. Skip the rest of the statements in the loop.
11. Define the hash KEY variable _N_ by executing the DEFINEKEY method when this variable is found.
12. Make all other variables including the hash key, _N_, data in the hash by executing the DEFINEDATA method. Each variable that we want to output will then be defined as data to the hash object _PDVHASH_.
13. End the loop started in line 6.
14. Execute the DEFINEDONE method to signal end of hash declaration.
15. End the loop started in line 3.
16. Read observations from SASHELP.CLASS and include the IN= data set option and the END= and NOBS= SET statement options. These variables will be output although they would not normally be.
17. Process with BY groups. Creates FIRST and LAST dot variables for each variable named. These automatic variables will also be output. They would not normally be.
18. Create a user defined variable with an assignment statement.
19. Update the hash object for each iteration of the data step using the ADD method. The value for the key, _N_, has been set by SAS and all other variables values have been assigned, read or otherwise populated. The simple ADD with no check for success is adequate in this example, because _N_ is unique for each observation.
20. On end of file DO the next three statements.
21. Execute the OUTPUT method, to write data in the hash _PDVHASH_ to a data set called “WORK.ALLVARSINPDV”. The output method simply writes all the data elements for the hash to the data set named by the data set argument.
22. Stop the data step.
23. End the DO started in line 20.
24. The ubiquitous RUN:

**OUTPUT: WORK.ALLVARSINPDV**

<table>
<thead>
<tr>
<th>F</th>
<th>I</th>
<th>L</th>
<th>F</th>
<th>I</th>
<th>L</th>
<th>A</th>
<th>I</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>R</td>
<td>A</td>
<td>S</td>
<td>S</td>
<td>R</td>
<td>A</td>
<td>_</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>H</td>
<td>W</td>
<td>S</td>
<td>S</td>
<td>T</td>
<td>S</td>
<td>S</td>
<td>E</td>
</tr>
<tr>
<td>C</td>
<td>e</td>
<td>e</td>
<td>T</td>
<td>.</td>
<td>T</td>
<td>T</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>n</td>
<td>N</td>
<td>i</td>
<td>i</td>
<td>.</td>
<td>N</td>
<td>N</td>
<td>.</td>
</tr>
<tr>
<td>O</td>
<td>a</td>
<td>e</td>
<td>e</td>
<td>o</td>
<td>a</td>
<td>S</td>
<td>A</td>
<td>g</td>
</tr>
<tr>
<td>b</td>
<td>s</td>
<td>n</td>
<td>b</td>
<td>m</td>
<td>e</td>
<td>g</td>
<td>h</td>
<td>h</td>
</tr>
<tr>
<td>s</td>
<td>s</td>
<td>d</td>
<td>s</td>
<td>e</td>
<td>x</td>
<td>e</td>
<td>e</td>
<td>e</td>
</tr>
</tbody>
</table>

| 1 | 1 | 0 | 19 | Alfred | M | 14 | 69.0 | 112.5 | 1 | 1 | 1 | 1 | 1 | 1 | 16.6115 | 0 | 1 |
| 2 | 1 | 0 | 19 | Alice | F | 13 | 56.5 | 84.0 | 1 | 0 | 1 | 1 | 1 | 1 | 18.4986 | 0 | 2 |
| 3 | 1 | 0 | 19 | Barbara | F | 13 | 65.3 | 98.0 | 0 | 0 | 1 | 1 | 1 | 1 | 16.1568 | 0 | 3 |
| 4 | 1 | 0 | 19 | Carol | F | 14 | 62.8 | 102.5 | 0 | 1 | 1 | 1 | 1 | 1 | 18.2709 | 0 | 4 |
| 5 | 1 | 0 | 19 | Henry | M | 14 | 63.5 | 102.5 | 1 | 0 | 1 | 1 | 1 | 1 | 17.8703 | 0 | 5 |
| 6 | 1 | 0 | 19 | James | M | 12 | 57.3 | 83.0 | 0 | 1 | 1 | 1 | 1 | 1 | 17.7715 | 0 | 6 |
| 7 | 1 | 0 | 19 | Jane | F | 12 | 59.8 | 84.5 | 1 | 0 | 1 | 1 | 1 | 1 | 16.6115 | 0 | 7 |
| 8 | 1 | 0 | 19 | Janet | F | 15 | 62.5 | 112.5 | 0 | 1 | 1 | 1 | 1 | 1 | 20.2464 | 0 | 8 |
| 9 | 1 | 0 | 19 | Jeffrey | M | 13 | 62.5 | 84.0 | 1 | 0 | 1 | 1 | 1 | 1 | 15.1173 | 0 | 9 |
| 10 | 1 | 0 | 19 | John | M | 12 | 59.0 | 99.5 | 0 | 1 | 1 | 1 | 1 | 1 | 20.0944 | 0 | 10 |
| 11 | 1 | 0 | 19 | Joyce | F | 11 | 51.3 | 50.5 | 1 | 0 | 1 | 1 | 1 | 1 | 13.4900 | 0 | 11 |
| 12 | 1 | 0 | 19 | Judy | F | 14 | 64.3 | 90.0 | 0 | 0 | 1 | 1 | 1 | 1 | 15.3030 | 0 | 12 |
| 13 | 1 | 0 | 19 | Louise | F | 12 | 56.3 | 77.0 | 0 | 0 | 1 | 1 | 1 | 1 | 17.0777 | 0 | 13 |
| 14 | 1 | 0 | 19 | Mary | F | 15 | 66.5 | 112.0 | 0 | 1 | 1 | 1 | 1 | 1 | 17.8045 | 0 | 14 |
| 15 | 1 | 0 | 19 | Philip | M | 16 | 72.0 | 150.0 | 1 | 0 | 1 | 1 | 1 | 1 | 20.3414 | 0 | 15 |
| 16 | 1 | 0 | 19 | Robert | M | 12 | 64.8 | 128.0 | 0 | 0 | 1 | 1 | 1 | 1 | 21.4297 | 0 | 16 |
| 17 | 1 | 0 | 19 | Ronald | M | 15 | 67.0 | 133.0 | 0 | 0 | 1 | 1 | 1 | 1 | 20.8285 | 0 | 17 |
| 18 | 1 | 0 | 19 | Thomas | M | 11 | 57.5 | 85.0 | 0 | 0 | 1 | 1 | 1 | 1 | 18.0733 | 0 | 18 |
| 19 | 1 | 1 | 19 | William | M | 15 | 66.5 | 112.0 | 0 | 1 | 1 | 1 | 1 | 1 | 17.8045 | 0 | 19 |

**CONCLUSION**

The purpose of this paper was to show how automatic SAS variables and other variables not otherwise output could be output using dynamic hash declaration and CALL VNEXT. Perhaps you will find using CALL VNEXT and the HASH object useful in a similar or different application.

**CONTACT INFORMATION**

John King
Ouachita Clinical Data Services, Inc.
1769 Highway 240 West
Caddo Gap, AR 71935
Email: ouachitaclinicaldataservices@gmail.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.