ABSTRACT
One of the often overlooked but extremely useful concepts in the SAS® language is the use of the shortcut notations, e.g., _numeric_, _character_, and, probably the most often used, _all_. While documentation on the use of these shortcuts is sometimes scarce, these notations can be used in a variety of ways to make the SAS programmer’s life easier. They can facilitate everything from data management to debugging programs and data analysis. In a flash, find bugs in the log file using put. Build arrays in a snap. Swiftly produce a list of contents used to manage your working library. This paper will address the use of shortcut notation in each of these areas and more and show how taking a shortcut is not only a quick fix, but a productive way to program.

INTRODUCTION
“The modern age has been characterized by a Promethean spirit, a restless energy that preys on speed records and shortcuts, unmindful of the past, uncaring of the future, existing only for the moment and the quick fix.”
Jeremy Rifkin
If the modern age brought us computer programming, then shortcuts were probably invented by programmers. If there’s a way to type two lines of code instead of three or one word instead of two, a programmer will find it. So, whether you’re an old programmer or new, you need to know about the shortcuts SAS has put into place for its programmers. There are some internal shortcuts for variable names that can be used in a variety of ways to facilitate nearly everything from data management to debugging programs and data analysis.

SHORTCUT NOTATION DEFINED
The internally defined shortcuts to list variables in SAS are as follows. This paper will focus on the last three, _numeric_, _character_, and _all_.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>xl-xn</td>
<td>Specifies variables X1 through Xn. The numbers must be consecutive.</td>
</tr>
<tr>
<td>x:</td>
<td>Specifies all variables that begin with the letter X.</td>
</tr>
<tr>
<td>x-a</td>
<td>Specifies all variables between X and A, inclusive. This notation uses the position of the variables in the data set.</td>
</tr>
<tr>
<td>x-numeric-a</td>
<td>Specifies all numeric variables between X and A, inclusive. This notation uses the position of the variables in the data set.</td>
</tr>
<tr>
<td>x-character-a</td>
<td>Specifies all character variables between X and A, inclusive. This notation uses the position of the variables in the data set.</td>
</tr>
<tr>
<td><em>numeric</em></td>
<td>Specifies all numeric variables.</td>
</tr>
<tr>
<td><em>character</em></td>
<td>Specifies all character variables.</td>
</tr>
<tr>
<td><em>all</em></td>
<td>Specifies all variables.</td>
</tr>
</tbody>
</table>

DEBUGGING
Perhaps the best known use of shortcut notation is using a put _all_; statement. This shortcut will give you a list in the log of all variables used in the data step, including the automated variables _N_, _ERROR_, FIRST_, LAST_, etc. on a single line. You could substitute put _numeric_; to only print out the numeric variables if you liked. Another use of this was demonstrated in Sample 307 by Rhoads on the http://sas.com website whereby you can output a single variable per line within an observation using the statement put (_all_); which will also omit the automated variables.
Example using `put _all_;`

```sas
data grocery;
  input SECTOR $ MANAGER $ DEPART $ SALES @@;
  datalines;
  SE 1 NP1 50 SE 1 P1 100 SE 1 NP2 120 SE 1 P2 80
  SE 2 NP1 40 SE 2 P1 300 SE 2 NP2 220 SE 2 P2 70
;
  data _null_;  
    set grocery;
    put _all_;
  run;
LOG OUTPUT
SECTOR=SE MANAGER=1 DEPART=NP1 SALES=50 _ERROR_=0 _N_=1
SECTOR=SE MANAGER=1 DEPART=P1 SALES=100 _ERROR_=0 _N_=2
SECTOR=SE MANAGER=1 DEPART=NP2 SALES=120 _ERROR_=0 _N_=3
```

Example using `put (_all_)`;

```sas
data _null_;  
  set grocery;
  put (_all_) (=/);
run;
LOG OUTPUT
SECTOR=SE MANAGER=1 DEPART=NP1 SALES=50
SECTOR=SE MANAGER=1 DEPART=P1 SALES=100
SECTOR=SE MANAGER=1 DEPART=NP2 SALES=120
```

**DATA MANAGEMENT**

**PROC CONTENTS**

If you ever need to see the entire contents of a SAS library, using the `_all_` notation is one way to get this information quickly. If you only want the list of datasets in the library, use the `NODS` option. By outputting the contents using an `(out=)` option, you can obtain a dataset containing variables with all the specifications of your datasets. From this you could take further steps to manage your data.

To output the library contents and output a SAS dataset (WORK.LIST1) containing a description of all variables in all datasets:

```sas
libname templ 'SAS-data-library';
proc contents data = templ._all_ out=list1;
run;
proc contents data = list1 ;
run;
```
The first 10 variables in the dataset WORK.LIST1:

-----Variables Ordered by Position-----

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Pos</th>
<th>Format</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LIBNAME</td>
<td>Char</td>
<td>8</td>
<td>152</td>
<td></td>
<td>Library Name</td>
</tr>
<tr>
<td>2</td>
<td>MEMNAME</td>
<td>Char</td>
<td>32</td>
<td>160</td>
<td></td>
<td>Library Member Name</td>
</tr>
<tr>
<td>3</td>
<td>MEMLABEL</td>
<td>Char</td>
<td>256</td>
<td>192</td>
<td></td>
<td>Data Set Label</td>
</tr>
<tr>
<td>4</td>
<td>TYPEMEM</td>
<td>Char</td>
<td>8</td>
<td>448</td>
<td></td>
<td>Special Data Set Type (From TYPE=)</td>
</tr>
<tr>
<td>5</td>
<td>NAME</td>
<td>Char</td>
<td>32</td>
<td>456</td>
<td></td>
<td>Variable Name</td>
</tr>
<tr>
<td>6</td>
<td>TYPE</td>
<td>Num</td>
<td>8</td>
<td>0</td>
<td></td>
<td>Variable Type</td>
</tr>
<tr>
<td>7</td>
<td>LENGTH</td>
<td>Num</td>
<td>8</td>
<td>8</td>
<td></td>
<td>Variable Length</td>
</tr>
<tr>
<td>8</td>
<td>VARNUM</td>
<td>Num</td>
<td>8</td>
<td>16</td>
<td></td>
<td>Variable Number</td>
</tr>
<tr>
<td>9</td>
<td>LABEL</td>
<td>Char</td>
<td>256</td>
<td>488</td>
<td></td>
<td>Variable Label</td>
</tr>
<tr>
<td>10</td>
<td>FORMAT</td>
<td>Char</td>
<td>8</td>
<td>744</td>
<td></td>
<td>Variable Format</td>
</tr>
</tbody>
</table>

To only see a listing of the library’s datasets:
Example using PROC DATASETS:

```
proc datasets memtype=data;
  contents data= temp1._all_ NODS;
run;
quit;
```

LOG OUTPUT

The CONTENTS Procedure

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Memtype</th>
<th>Size</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GROCERY</td>
<td>DATA</td>
<td>5120</td>
<td>06FEB2006:08:56:49</td>
</tr>
<tr>
<td>2</td>
<td>LIST1</td>
<td>DATA</td>
<td>17408</td>
<td>06FEB2006:09:00:04</td>
</tr>
</tbody>
</table>

REMOVE FORMATS AND INFORMATS
Sometimes it is preferable to view data without their assigned formats and informats. Using the _all_ shortcut with a format or informat statement can make is easy to temporarily or permanently remove the formats and informats.

Temporary solution
An example may be a simple PRINT procedure without formats. The nice thing about this is that you can remove formats for all variables even if you specify only a few in the var statement. These formats are not removed from the dataset itself, only from the printed data.

```
proc print data = temp1.tablenew;
  var PT SEX DOB;
  format _all_;
run;
```

Permanent solution
When importing data from unknown sources, sometimes the original format and informats placed on the dataset is not desired. However, by deleting the formats and infomats using _all_, your problem can be solved. Here is an example of importing an Excel spreadsheet and permanently deleting all formats and informats.

```
libname templ 'SAS-data-library';

proc import datafile="c:/table.xls" out= templ.tablenew;
run;

data templ.tablenew;
  set templ.tablenew;
  format _all_;
  informat _all_;
run;
```
A second permanent way to change data set attributes would be to use the DATASETS procedure in SAS. It is possible to remove variable labels and formats accompanied by the \_all\_ shortcut. For example, the following code will remove the labels and formats from the WORK.LIST1 dataset.

```sas
proc datasets lib=work;
  modify list1;   /* modify the data set work.list1 */
  attrib _all_ label=' ';  /* remove all labels */
  attrib _all_ format=;  /* remove all formats */
run;
quit;
```

OUTPUT: the labels and formats have been stripped from WORK.LIST1.

### The CONTENTS Procedure

####-----Variables Ordered by Position-----

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LIBNAME</td>
<td>Char</td>
<td>8</td>
<td>152</td>
</tr>
<tr>
<td>2</td>
<td>MEMNAME</td>
<td>Char</td>
<td>32</td>
<td>160</td>
</tr>
<tr>
<td>3</td>
<td>MEMLABEL</td>
<td>Char</td>
<td>256</td>
<td>192</td>
</tr>
<tr>
<td>4</td>
<td>TYPEMEM</td>
<td>Char</td>
<td>8</td>
<td>448</td>
</tr>
<tr>
<td>5</td>
<td>NAME</td>
<td>Char</td>
<td>32</td>
<td>456</td>
</tr>
<tr>
<td>6</td>
<td>TYPE</td>
<td>Num</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>LENGTH</td>
<td>Num</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>VARNUM</td>
<td>Num</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>LABEL</td>
<td>Char</td>
<td>256</td>
<td>488</td>
</tr>
<tr>
<td>10</td>
<td>FORMAT</td>
<td>Char</td>
<td>8</td>
<td>744</td>
</tr>
</tbody>
</table>

### DROP AND KEEP STATEMENTS

You can also use these shortcuts in a drop or keep statement at the end of your data step only to keep the numeric or character variables. Using the \_all\_ in a drop or keep statement works too, but it might be a stretch to come up with a legitimate use!

```sas
keep _numeric_
```

### LIBNAMES AND FILENAMES

Used with clear and list statements, the \_ALL\_ shortcut notation will allow you prompt control over user defined libnames and filenames.

To list all libnames or filenames in the log:

```sas
libname _all_ list;
```

OR

```sas
filename _all_ list;
```

To clear all user-defined libname or filename definitions. Note that system-defined libnames and filenames are not affected.

```sas
libname _all_ clear;
```

OR

```sas
filename _all_ clear;
```

### OUTPUT DELIVERY SYSTEM (ODS)

To close all ODS output:

```sas
ods _all_ close;
```
DATA ANALYSIS

ARITHMETIC FUNCTIONS
These shortcuts can be used in a variety of ways to help you quickly analyze data. For example, use the `numeric` shortcut in any arithmetic function that accepts a list of variables such as `mean(of numeric)` and `max(of numeric)`. A list of functions where it is possible to use `<function>(of NUMERIC)` follows.

**TABLE 2. ARITHMETIC FUNCTIONS ACCEPTING SHORTCUT NOTATION:**

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>sample size</td>
</tr>
<tr>
<td>SUMWGT</td>
<td>sum of the weights</td>
</tr>
<tr>
<td>MEAN</td>
<td>sample mean</td>
</tr>
<tr>
<td>SUM</td>
<td>sum of the observations</td>
</tr>
<tr>
<td>STD</td>
<td>standard deviation</td>
</tr>
<tr>
<td>VAR</td>
<td>variance</td>
</tr>
<tr>
<td>SKEWNESS</td>
<td>skewness</td>
</tr>
<tr>
<td>KURTOSIS</td>
<td>kurtosis</td>
</tr>
<tr>
<td>MAX</td>
<td>largest value</td>
</tr>
<tr>
<td>MIN</td>
<td>smallest value</td>
</tr>
<tr>
<td>NOBS</td>
<td>number of observations</td>
</tr>
<tr>
<td>RANGE</td>
<td>range</td>
</tr>
<tr>
<td>MODE</td>
<td>most frequent value</td>
</tr>
<tr>
<td>NMISS</td>
<td>number of missing values</td>
</tr>
<tr>
<td>USS</td>
<td>uncorrected sum of squares</td>
</tr>
<tr>
<td>CSS</td>
<td>corrected sum of squares</td>
</tr>
<tr>
<td>CV</td>
<td>coefficient of variation</td>
</tr>
<tr>
<td>STDERR</td>
<td>standard error of the mean</td>
</tr>
</tbody>
</table>

PROCEDURES
Use the shortcuts with many statistical procedures to make your descriptive and modeling statistics a breeze. Supported procedures include CORR, FREQ, MEANS, SUMMARY, TABULATE, UNIVARIATE, ANOVA, GLM, etc. Note: You cannot use shortcuts to list variable names in the INDEX CREATE statement in PROC DATASETS.

ARRAYS
Use the shortcuts to speedily build arrays and analysis variables. For example, build an array using the `numeric` shortcut to use the numeric date variables in your dataset to compute the number of days between each visit and the date of consent.
Example of using _NUMERIC_ with an array:

```sas
data x;
  set file1;
  array numx(5) _numeric_; /*all visit dates */
  array dayx(5) dayct1-dayct5;
  do i = 1 to 5;
    dayx(i) = consent_date-numx(i);
  end;
run;
```

**CONCLUSIONS**

While the shortcut notations are simple in concept, the applications for them are widespread and perhaps underutilized. This paper certainly has not exhausted their utility, but hopefully clarified that they are an essential tool for programmers, the foremost architect of the shortcut.

**REFERENCES**


**CONTACT INFORMATION**

Robby Diseker  
PPD, Inc.  
3151 South 17th St.  
Wilmington, NC 28412  
Work Phone: 910-772-2305  
Fax: 919-654-0726  
Email: Robert.Diseker@wilm.ppdi.com

An electronic version of this program can be obtained via e-mail from the author.

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.