

# SAS®'s Various Varying Variables, or 1000+ Ways to Manipulate SAS Variables

## SAS®'s Various Varying Variables, or 1000+ Ways to Manipulate SAS Variables

Paul A. Choate  
California State Developmental Services



---

---

---

---

---

---

---

---



### Part 1 Overview

#### PART 1: VARIABLE ATTRIBUTES

- Definition of Variable Attributes and the Program Data Vector
- Changing Variable Attributes with Data Step Statements
- Indexing Variables, Changing Variable Types, Reordering the PDV
- Changing Variable Attributes with Procedures
- Variable Information Functions and Call Routines

---

---

---

---

---

---

---

---



### Part 2 Overview

#### PART 2: VARIABLE CONTENTS AND HANDLING

- Tokens, Name Literals, and Special Constants
- Number of Variables and Variable Size Limits
- Variable Lists in Functions, Statements, Options, and Arrays
- Variable Comparisons

---

---

---

---


---

---

---

---

# SAS®'s Various Varying Variables, or 1000+ Ways to Manipulate SAS Variables



## Eight Attributes of Data Step Variables

- **Name**
  - Up to 32 characters, first character letter or underscore (A, z, \_)
  - Other characters can be mixed case letters, digits, or underscores
  - Processed as uppercase
- **Type** - Numeric or character
- **Length** -
  - Character: 1 to 32,767 bytes – default 8
  - Numeric: 2 or 3 to 8 bytes – default 8
- **Format** - Print characters or machine form, default BEST12. for numeric, Sw. for character
- **Informat** - Character or numeric w.d for numeric, Sw. for character
- **Label** - Up to 256 characters
- **Position in PDV** - 1- n
- **Index type** - None, simple, composite, or both

---

---

---

---

---


---

---

---

---

---



## The Program Data Vector

Know your PDV!

Attributes		Data Set Variables				Automatic Variables		
Descriptor	Name	Name	Sex	Age	Height	Weight	._ERROR	._N
	Type	char	char	num	num	num	num	num
	Length	8	1	8	8	8	8	8
	Format	\$8	\$1	best12.	best12.	best12.	best12.	best12.
	Informat	\$8	\$1	12	12	12	12	12
	Label							
	Position	1	2	3	4	5	6	7
	Index type	none	none	none	none	none	none	none
Flags	Drop flag	no	no	no	no	no	yes	yes
	Retain flag	yes	yes	yes	yes	yes	yes	yes
Values	Alfred	M	14	69	112.5		0	1

---

---

---

---

---


---

---

---

---

---



## Changing Variable Attributes - Data Step Statements

Five of the eight attributes can be directly changed with data step statements

1. RENAME Statement & Options
2. LENGTH Statement
3. FORMAT Statement
4. INFORMAT Statement
5. LABEL Statement

- ATTRIB Statement (all above except RENAME)

---

---

---

---

---

---

---

---

---

---



## RENAME Statement

Available in DATA, SET, MERGE, MODIFY, UPDATE statements

```
data class(rename=(Sex=Gender));  
  *sex renamed after processing;  
  set sashelp.class(rename=(Age=Age_Yrs));  
  *age renamed before processing;  
  Rename Name=First_Name;  
  *name renamed after processing;  
run;
```

Output of SAS procedures including PROC SQL:

```
proc sql;  
  create table class (rename=(Sex=Gender))  
  as select *  
  from sashelp.class (rename=(Age=Age_Yrs));  
quit;
```

---

---

---

---

---

---

---

---



## LENGTH Statement

Statement location matters!

```
*defined as $6 and first variable in PDV;  
*length changed from $1 to $6;  
data class;  
  length sex $6;  
  set sashelp.class;  
  if sex='F' then sex='Female';  
run;
```

```
*no effect when placed after set statement;  
*sex remain $1 and same position as input;  
data class;  
  set sashelp.class;  
  length sex $6;  
  if sex= 'F' then sex='Female';  
run;
```

---

---

---

---

---

---

---

---



## INFORMAT & FORMAT Statements

Statement location affects length & PDV order

```
*format & informat changed regardless of  
statement location in data step;
```

```
data class;  
  set sashelp.class;  
  informat sex $6.;  
  format sex $6.;  
run;
```

```
*when before set statement variable is defined  
in first pdv position and length is set to $6;
```

```
data class;  
  informat sex $6.;  
  format sex $6.;  
  set sashelp.class;  
run;
```

---

---

---

---

---

---

---

---



## LABEL Statement

Works anywhere, but location changes PDV order  
(like INFORMAT and FORMAT)

```
*new label w/ original position;
data class;
  set sashelp.class;
  label age='Age of Student';
run;
```

```
*new label w/ new position;
data class;
  label age='Years of Age';
  set class;
run;
```

---

---

---

---

---

---

---

---



## ATTRIB Statement

ATTRIB changes lengths, formats, informats,  
and/or labels  
(& location has same side-effects on attributes)

```
data class1;
  attrib sex label ='Gender'
         length =$6;
  set sashelp.class;
run;
```

```
data class2;
  set sashelp.class;
  attrib sex label ='Gender'
         length =$6;
run;
```

---

---

---

---

---

---

---

---



## Indexing Variables - 1 of 3

Indexing with Data Step  
DATA Statement Option

```
data class(index=(sex agename=(age name)));

  *simple and composite indexes;

  set sashelp.class;

run;
```

---

---

---

---

---

---

---

---



## Indexing Variables - 2 of 3

Indexing with SQL Procedure Statements  
CREATE or CREATE INDEX

```
proc sql; *build index during creation;
create table
  class(index=(sex agetname=(age name))) as
  select *
  from sashelp.class;
quit;

*or;

proc sql;
create table class as
select *
from sashelp.class;
create index sex on class (sex);
*build index on pre-existing data set;
quit;
```

---

---

---

---

---

---

---

---



## Indexing Variables - 3 of 3

Indexing with DATASETS Procedure

```
data class;
set sashelp.class;
run;

proc datasets library=work nolist;
modify class;
index create age;
*build index on pre-existing data set;
quit;
```

---

---

---

---

---

---

---

---



## Changing Variable Type

Variable Type Change  
(with Sleight of Hand)

```
data class(drop=_Age);

  retain Name Sex Age Height Weight;
  *maintains PDV order;

  set sashelp.class(rename=(Age=_Age));
  *rename variable;

  Age=put(_Age,z2.);
  *transform variable to new type;

run;
```

---

---

---

---

---

---

---

---



### Reordering PDV with RETAIN Statement

The RETAIN statement is redundant for:

- variables defined by SET, MERGE, MODIFY or UPDATE statements,
- variables defined by statement options in SET, MERGE, MODIFY, UPDATE, FILE and INFILE statements,
- a sum statement (i.e. count+1; not sum function),
- initialized array statement variables

---

---

---

---

---

---

---

---



### Reordering PDV with RETAIN Statement

```
data class;  
    retain Age Height NewVar Weight Name Sex;  
    NewVar=.;  
    *set NewVar to missing to negate RETAIN;  
    set sashelp.class;  
    *... calculate NewVar ...;  
run;
```

---

---

---

---

---

---

---

---



### Accessing and Changing Variable Attributes with Procedures

Three procedures to access or  
change variable attributes

- PROC DATASETS
- PROC CONTENTS
- PROC SQL

---

---

---

---

---

---

---

---



## Changing Variable Attributes with PROC DATASETS

```
proc datasets library=work nolist;
  modify class;
  format height 5.2;
  informat height 6.3;
  index create age;
  *only INDEX statement causes file processing;
  rename age=AgeYrs;
  *RENAME also renames index;
  label AgeYrs='Years of Age';
quit;
```

---

---

---

---

---

---

---

---



## Accessing Variable Attributes with PROC CONTENTS

(OR PROC DATASETS CONTENTS)

- Reports the variable descriptor information
- Cannot change variable descriptor information
- Efficient, only reads file header
- Useful for examining foreign data sets
- PROC CONTENTS can read sequential files  
i.e. Tape or an FTP server (PROC DATASETS cannot)

---

---

---

---

---

---

---

---



## Changing Variable Attributes with PROC SQL Column-Modifiers

```
*change Sex to Gender and modify descriptor;
proc sql;
  create table class as
    select Name,
           Sex as Gender length=6 format=$6.
           label='Student Gender',
           Age, Height, Weight
    from sashelp.class;
quit;

*or rename option;
from sashelp.class(rename=(Sex=Gender));
```

---

---

---


---

---

---

---

---



## Accessing Variable Attributes with PROC SQL dictionary.columns

```
proc sql noprint;
  create table class_descriptor as
  select *
  from dictionary.columns
  where libname = 'SASHELP' and
         memname = 'CLASS';
quit;
```

class_descriptor							
name	type	length	npos	varnum	label	format	informat
Name	char	8	24	1			
Sex	char	1	32	2			
Age	num	8	0	3			
Height	num	8	8	4			
Weight	num	8	16	5			

---

---

---


---

---

---

---

---



## Version 9 Variable Information Functions and Call Routines

Including: VFORMAT, VINFORMAT, VLABEL,  
VLENGTH, VNAME, VTYPE, and VVALUE

```
data _null_;
  if 0 then set sashelp.class;
  *don't read data set - load PDV only;
  vlen=vlength(name);
  put "length of name is " vlen;
  stop;
run;
```

length of name is 8

---

---

---


---

---

---

---

---



## Name Literals

VALIDVARNAME=ANY required for variable name  
literals, but not for special dbms table names

```
options validvarname=any;
libname msacc access 'Data.mdb' ;
data msacc.'new class data'n ;
*validvarname not required for dbms table name;
  set sashelp.class;
  rename name='very-long #name#n';
*validvarname required for name;
run;
libname msacc clear;
options validvarname=v7;
```

---

---

---

---

---

---

---

---



## Overview Part 2

### PART 2: VARIABLE CONTENTS AND HANDLING

- Tokens, Name Literals, and Special Constants
- Number of Variables and Variable Size Limits
- Variable Lists in Functions, Statements, Options, and Arrays
- Variable Comparisons

---

---

---

---

---

---

---



## SAS Language Tokens - 1 of 4

The SAS programming language is composed of four types of tokens:

- Name tokens
- Literal tokens
- Number tokens
- Special character tokens

Several parsing rules apply, such as multiple blanks and line breaks aren't read, thus embedded blanks and line breaks do not affect processing.

---

---

---

---

---

---

---



## SAS Language Tokens - 2 of 4

**Name tokens** are strings of letters, digits, and underscores and not beginning with a digit.

Name tokens are used to name variables, data sets, arrays, statements, functions, procedures, librefs and filerefs, and other SAS language elements.

Mixed case letters are not distinguished, name tokens are processed as uppercase.

Name token types have special length requirements. For example, variables name tokens are limited to 32 characters, while libref name tokens can be up to 8 characters.

---

---

---

---

---

---

---

# SAS®'s Various Varying Variables, or 1000+ Ways to Manipulate SAS Variables



## SAS Language Tokens - 3 of 4

Name tokens include name literals

**Name literals** are quoted strings that may include special characters, followed by the letter n.

Name literals are used only in variables, statement labels, and DBMS column and table names. They are accessible in the DATA step and PROC SQL step.

Except for DBMS table names, SAS name literals require the VALIDVARNAME=ANY option.

---

---

---

---

---

---

---

---



## SAS Language Tokens - 4 of 4

**Special character tokens** are characters that are not letters, digits, or underscores.

Special characters include = + - % & : ( ) ;.

**Literal tokens** are quoted character strings and may include special characters and spaces.

**Number tokens** are numeric strings that may contain some special characters, the letters e and x, or date and time constants.

---

---

---

---

---

---

---

---



## Number of Variables

Starting with SAS 9.1, the maximum number of variables is greater than 32,767 and is dependent on environmental and file attributes

```
data wide(drop=i);  
  array _[1000000] 8 ;  
  do i = 1 to 1e6;  
    _(i)=ranuni(0);  
  end;  
run ;
```

Variables \_1, \_2, \_3, ... \_1000000

---

---

---

---

---

---

---

---

# SAS®'s Various Varying Variables, or 1000+ Ways to Manipulate SAS Variables



## Size of Numeric Variables

The largest exact integer is found with the function  
CONSTANT('EXACTINT' <, nbytes>)

On XP EXACTINT for eight bytes is  
9,007,199,254,740,992

The largest representable integer is found with the  
function CONSTANT('BIG')

On XP SAS reports BIG=1.7976931348623E308

It can be shown to be exactly  
1.79769313486231580849E308-1

---

---

---

---

---

---

---

---



## Numeric Variable Space Considerations

Numeric categorical variables, such as Likert scores,  
ZIP codes, and ID numbers can be stored as either  
numeric or character type variables. Space and I/O  
considerations can be significant.

For large integral values of 15 digits, storing as eight  
byte numeric saves nearly twice as much space as  
15 byte character, and will be precise up through  
EXACTINT8.

For a single digit scale of 0-9, one byte of character  
storage is eight times smaller than the default 8 byte  
numeric, and three times smaller than the smallest  
three byte numeric storage.

---

---

---

---

---

---

---

---



## Size of Numeric Variables - Dates

```
data _null_;  
  x=constant('EXACTINT',4);  
  put 'largest exact 4 byte integer= ' x /  
      'works as a date= ' x date9. /  
      'or as a time= ' x datetime20. ;  
  x=constant('EXACTINT',6);  
  put 'largest exact 6 byte integer= ' x /  
      'works as a datetime= ' x datetime20. ;  
run;  
largest exact 4 byte integer= 2097152  
works as a date= 23OCT7701  
or as a time= 25JAN1960:06:32:32  
largest exact 6 byte integer= 137438953472  
works as a datetime= 09APR6315:15:04:32
```

---

---

---

---

---

---

---

---

# SAS®'s Various Varying Variables, or 1000+ Ways to Manipulate SAS Variables



## Character Variables: Size and Hexadecimal Characters

- Up to 32,767 characters (ten typed pages)
- Stored as processed (except trimmed trailing blanks)
- Trailing blanks implied by variable length
- May include non-printing characters (tabs or CRLF)
- Platform dependent - blanks are ASCII '20'x or EBCDIC '40'x
- Hexadecimal literals, formats, functions (i.e. RANK)
- \$CHARw. informat preserves leading blanks

---

---

---

---

---

---

---

---



## Special Numeric Constants

Notation	Value	Special Constant
Scientific	-32.5	-3.25e1
Hexadecimal	23	17x
Dates	0	'01Jan1960'd
Time	0	'12:00:00am't
Datetime	0	'01Jan1960:12:00:00am'dt

```
data constants;
  numvar = 17x + -3.e-1 + '12:01:40am't;
run
```

numvar = 23 + -0.3 + 100 = 122.7

---

---

---

---

---

---

---

---



## Special Character Constants

Notation	Value	Special Constant
Embedded Quotes	We'll	"We'll" or 'We'll'
Hexadecimal	We'll	'5765276C6C'x

- Quoted strings of up to 32,767
- To assign character variables and other data step tasks

```
data constants;
  equalstrings =
    ('We'll' = '57,65,27,6C,6C'x);
run;
```

Optional commas in hexadecimal character constants

---

---

---


---

---

---

---

---



### Variable Lists

Variable List Type	Notation
Standard List	VarA, VarB, VarC, ..., VarZ
Special ALL	_all_
Name Range	VarA--VarZ
Special Numeric	_numeric_
Numeric Name Range	VarA -numeric- VarZ
Special Character	_character_ or _char_
Character Name Range	VarA -character- VarZ or VarA -char- VarZ
Numbered Range	Var1-VarN
Name Prefix	Var:

---

---

---


---

---

---

---

---



### Lists in Functions

```
data lists;  
  set sashelp.class;  
  sum=sum(of _numeric_);  
  *note the "of" operator;  
  length cat $10;  
  cat=cat(of _character_);  
run;
```

---

---

---


---

---

---

---

---



### Lists in Statements

```
data list;  
  input (var1-var3) (2.,+1);  
  *note input's special list syntax "()" ;  
  rename var1-var3=Item1-Item3;  
  *rename only accepts numbered range;  
datalines;  
  ;  
proc freq data=sashelp.class;  
  tables _character_;  
proc sort data=sashelp.class out=class;  
  by name-char-height;  
run;
```

---

---

---

---

---

---

---

---

# SAS®'s Various Varying Variables, or 1000+ Ways to Manipulate SAS Variables



## Lists in Options

```
data lists(keep=.);  
  *keeps only the two renamed variables;  
  set sashelp.class(keep=_numeric_);  
  rename weight=_weight  
         height=_height;  
run;
```

---

---

---

---

---

---

---

---



## Lists in Arrays

```
data class(drop=.);  
  set sashelp.class;  
  
  array nums _numeric_;  
  array chars $ _char_;  
  
  do _i = 1 to dim(nums);  
    if nums(_i)=0 then nums(_i)=.;  
  end;  
  
  do _j = 1 to dim(chars);  
    if anydigit(chars(_j)) then chars(_j)='';  
  end;  
run;
```

---

---

---

---

---

---

---

---



## Truncated Character Variable Comparisons in the Data Step

Works with any comparison operator –  
EQ:, NE:, GT:, LT:, GE:, LE:, or IN:

```
data class;  
  set sashelp.class;  
  if name =: 'Ja' or  
     'P' LE: name LT: 'T';  
run;  
  
*Selects names starting with Ja, P, Q, R, & S;  
  
*Note sort order - ASCII 'A'<'a'  
                  EBCDIC 'a'<'A';
```

---

---

---

---

---

---

---

---



### Truncated Character Variable Comparisons in SQL

Equivalent to Data Step Colon Operator—  
EQT, NET, GTT, LTT, GET, LET, *but not INT*

```
proc sql;  
  create TABLE class as  
  select *  
  from sashelp.class  
  where name EQT "Ja" or  
  ('P' LET name and name LTT 'T');  
quit;  
  
*Selects names starting with Ja, P, Q, R, S;  
*('P' LET name LTT 'T') not permissible in SQL;
```

---

---

---

---

---

---

---

---



### IN Operator Colon for Integer Lists

```
*test for integers 13, 14, 15, 16;  
  
data class;  
  set sashelp.class;  
  if age in (13:16);  
run;
```

---

---

---

---

---

---

---

---



### Acknowledgements

Thanks to Section Chairs  
**Kirk Lafler** and  
**Patrick Thornton**  
for all their hard work in organizing  
Data Presentation and Business  
Intelligence!

---

---

---

---

---

---

---

---

# SAS®'s Various Varying Variables, or 1000+ Ways to Manipulate SAS Variables



## Contact Information

Your comments and questions are valued and encouraged. Feel free to contact the author at:

Paul Choate, Senior Programmer Analyst  
California Department of Developmental Services  
Phone: (916) 654-2160  
E-mail: [Paul.Choate@dds.ca.gov](mailto:Paul.Choate@dds.ca.gov)

*Join the SAS-L @ [LISTSERV.UGA.EDU](mailto:LISTSERV.UGA.EDU) or Google Groups!*

---

---

---

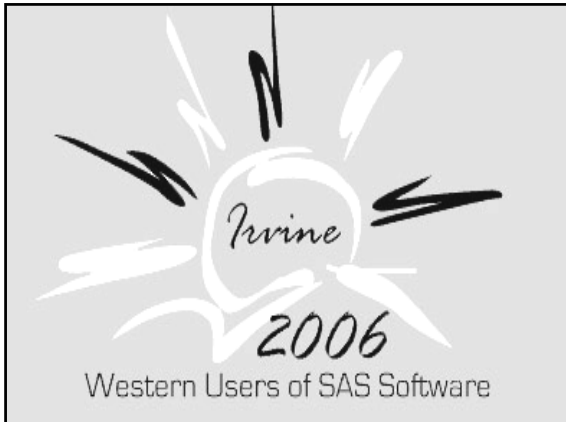
---

---

---

---

---



---

---

---

---

---

---

---

---