

Why Does SAS® Run Clockwise?

Rob Krajcik

Bristol-Myers Squibb Company

HASUG Feb 2009

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 registered trademarks or trademarks of their respective companies.

Acknowledgements

Many of the quirks, traps, and oddities presented here were probably recycled from other SAS® user group papers and presentations.

I want to thank Jerry Le Breton (SAUSAG), Phil Mason, Ron Fehd, Ian Whitlock, Bob Virgile, Paul Dorfman, Michael Davis, Art Carpenter, Malachy Foley, Richard DeVenezia, Frank DiLorio, Rick Langston, Neil Howard, Ray Pass, Kirk Paul Lafler, and many others, most of whom unknowingly gave me ideas for this presentation.

I have a list of websites for further reading in the back

Introduction

By "Why Does SAS Run Clockwise" I mean,
Why does SAS behave the way it does?

Q: Why do clocks run Clockwise?

A: Mechanical clocks with hands were originally built to imitate the path of a sundial shadow. In the northern hemisphere, where sundials were first used extensively, the shadow on a sundial moves around its face in a clockwise direction, as the sun moves across the sky.

Introduction

Q: What is a Quirk?

A: peculiarity of behavior; an idiosyncrasy

To a logician, the following statements may seem quirky or even impossible:

$$A = \text{not } (A)$$

$$X = X + 1$$

When you understand the equals sign is really an assignment statement, they don't seem so quirky.

Introduction

The BASE SAS software is comprised of three major parts: the Data Step, Proc (procedure) Steps, and a macro language. There is also a Screen Control Language (which later became the Software Component Language).

Every language has its quirks, and SAS is no exception. These are not bugs or software defects, just things that seem to act in a non-intuitive way. You may hear speakers talk of SAS "traps" and "gotchas".

These are the same as what I am talking about here. Some of these come about from programmers who have an intuitive feel for how SAS works, but perhaps hadn't read the manual in detail. Perhaps it was undocumented.

As an experienced PL/I programmer new to SAS in 1979, many features of the SAS language seemed a bit quirky to me. Maybe they won't to you.

OK, Here's an example:

Introduction

Q: Why does SAS use the BEST12. format as a default numeric to character conversion?

A The default character length is 12 bytes.

B It offered the best numerical accuracy.

C It seemed like the "best" thing to do at the time.

Introduction

It might also help to understand how SAS (both the software and the company) evolved over time. So I digress for a moment...

SAS History

Q: Which mainframe was SAS was first designed to run on?

A Univac

B IBM

C PR1ME

D Digital

SAS History

Q: When SAS first entered the market in 1976 it had many competitors including SPSS, BMDP, OSIRIS, Omnitab and P-Stat. One feature SAS had that over its competitors was:

- A Use of double-precision for calculations.
- B A means of cleaning and manipulating the data.
- C Readable documentation.
- D Customer Support.

SAS History

Q: The SAS data step language most closely resembles:

A COBOL (CODASYL Committee, 1960 -- and Grace Hopper)

B PL/I (George Radin, IBM, 1964)

C FORTRAN (John Backus, IBM, 1954-57)

D Algol (J. Backus and P. Naur, 1958-60)

SAS History

Q: In which release of SAS below was the macro language changed from statements like:

```
macro counter x++1; %
```

to:

```
%macro counter();  
%let x=%eval(&x+1);  
%mend;
```

A SAS V4

B SAS 82.3

C SAS V6.06

D SAS V5.08

SAS History

Q: Which release of SAS introduced things like the "In" Operator, "Where" Processing, Stored Compiled Data Step Code, Proc Sql, Indexing of SAS Data Files, and the "File Name" Statement?

A SAS V4

B SAS 82.3

C SAS V6.06

D SAS V5.18

SAS History

Q: Many SCL (and data step) functions became available to the macro processor via %sysfunc in which release?

A SAS V6.03

B SAS 82.3

C SAS V6.09E

D SAS V5.18

SAS History

Q: The Output Delivery System (ODS) and long variable names were first introduced in which release?

A SAS V6.12

B SAS V7.1

C SAS V6.09E

D SAS V5.18

SAS History

Q: Which release of SAS introduced a whole host of new character functions, date formats, many ODS enhancements, longer names for formats and informats, a hash object that allows functionality similar to the MERGE statement without sorting data or building formats, and PERL regular expressions?

A SAS V6.12

B SAS V7.1

C SAS V8.2

D SAS V9.1

That's How it Quirks

```
data _null_;  
    put _all_;  
run;
```

```
_ERROR_=0 _N_=1
```

```
NOTE: DATA statement used (Total process time):
```

Q: Where did `_ERROR_` and `_N_` come from?

A: Two automatic variables are created by every DATA step: `_N_` and `_ERROR_`. The value of `_N_` represents the number of times the DATA step has iterated. `_ERROR_` is 0 by default but is set to 1 whenever an error is encountered,

That's How it Quirks

```
data _null_;  
  do i=1 to 3;  
    x++1; /* or x+(+1) or x+1 */  
    y+-1; /* or y+(-1) or retain m1 -1; y+m1; */  
  end;  
  put (i x y) (=);  
run;
```

Q: What is "x++1" shorthand for?

A: retain x; x = sum(x,1);

Q: What's the value of i? of x? of y?

A: i=4 x=6 y=-6

That's How it Quirks

```
data _null_;  
    if done then put (_all_)(=);  
    cnt=_n_;  
    set sashelp.class(obs=5) end=done;  
run;
```

done=1

NOTE: There were 5 observations read from the data set SASHELP.CLASS

Q: What does "put (_all_)..." do? Why do I see only "done=1" on the log?

A: Where "put _all_;" is a command to dump the LPDV in the form (var=value), the "_all_" in "put (_all_) (=)" is a list. It is a list of all variables in the LPDV at this point in the compilation of the program. Automatic variables are not included in this list.

Q: What does obs= do?

A: Reads to that LAST observation (It is NOT the number of obs to read). Think lastobs.

That's How it Quirks

```
data _null_;  
    put 'Before' _n_=i;  
    set sashelp.class(obs=3) end=done;  
    put 'After ' _n_=i;  
run;
```

Before _N_=1

After _N_=1

Before _N_=2

After _N_=2

Before _N_=3

After _N_=3

Before _N_=4

NOTE: There were 3 observations read from the data set SASHELP.CLASS.

Q: Why did the log print "Before _N_=4" but not "After _N_=4" ?

A: The data step with an implied loop ends when it tries to read beyond the end of the input dataset (on a SET or INPUT statement).

Q: And where is the better place to put an 'if done then ...' statement?

That's How it Quirks

```
data _null_;  
    if 0 then set sashelp.class nobs=nobs;  
    put nobs=;  
    stop;  
run;  
nobs=19  
NOTE: DATA statement used (Total process time):
```

Q: Did the "set" statement execute?

A: No.

Q: How did we get nobs=19 on the saslog?

A: The SAS compiler inserted the value into the variable.

That's How it Quirks

```
data _null_;  
    set sashelp.class(where=(age ge 15));  
    put (name age height) (=);  
run;
```

```
Name=Janet Age=15 Height=62.5  
Name=Mary Age=15 Height=66.5  
Name=Philip Age=16 Height=72  
Name=Ronald Age=15 Height=67  
Name=William Age=15 Height=66.5
```

```
NOTE: There were 5 observations read from the data set SASHELP.CLASS.  
      WHERE age>=15;
```

Now add firstobs and obs...

That's How it Quirks

```
data _null_;  
    set sashelp.class(where=(age ge 15) firstobs=2 obs=9);  
    put (name age height) (=);  
run;
```

Name=Mary Age=15 Height=66.5

Name=Philip Age=16 Height=72

Name=Ronald Age=15 Height=67

Name=William Age=15 Height=66.5

NOTE: There were 4 observations read from the data set SASHELP.CLASS.
WHERE age>=15;

Q: What happened? Why only 4 observations?

A: WHERE Processing takes precedence over firstobs and (last)obs. Here it found five observations matching the where clause started at the second one and would have gone to the ninth (for a total of 8) if there were that many matching the WHERE clause.

That's How it Quirks

```
data males(keep=_n_ name);  
  do j=1 by 1 until(sex='M');  
    set sashelp.class(firstobs=3 obs=11);  
    i++1;  
    put i= j= _n_ name= sex=;  
  end;
```

```
run;
```

```
i=1 j=1 _N_=1 Name=Barbara sex=F  
i=2 j=2 _N_=1 Name=Carol sex=F  
i=3 j=3 _N_=1 Name=Henry sex=M  
i=4 j=1 _N_=2 Name=James sex=M  
i=5 j=1 _N_=3 Name=Jane sex=F  
i=6 j=2 _N_=3 Name=Janet sex=F  
i=7 j=3 _N_=3 Name=Jeffrey sex=M  
i=8 j=1 _N_=4 Name=John sex=M  
i=9 j=1 _N_=5 Name=Joyce sex=F
```

NOTE: There were 9 observations read from the data set SASHELP.CLASS.

NOTE: The data set WORK.MALES has 4 observations and 1 variables.

Questions on next slide...

That's How it Quirks

```
data males(keep=_n_ name);  
  do j=1 by 1 until(sex='M');  
    set sashelp.class(firstobs=3 obs=11);  
    i++1;  
    put i= j= _n_= name= sex=;  
  end;  
run;
```

Q: When is the until statement evaluated?

A: At the end of the do loop (while is evaluated at the beginning).

Q: How many records are we reading from sashelp.class?

A: 9. (last)obs - firstobs + 1

Q: How many records were written to work.males? Why?

A: 4. We were looping until sex='M', the implied output statement at the bottom of the data step only saw males. At `_N_=5` it read the 9th record and stopped.

Q: Which variable was not kept on the dataset males? Why?

A: `_N_`. Automatic variables are NEVER written to output data sets.

That's How it Quirks

```
proc format;  
    value result  
    0-65 = 'Fail'  
    66-100='Pass'  
    ;  
quit;  
data _null_;  
    score=65.5;  
    grade=put(score,result.);  
    put grade=;  
run;
```

r a "Fail"?

A: 65.5

Q: And how do we fix it?

A: One way is to use the "<" symbol to exclude 66 from the first range.

That's How it Quirks

```
%let x=abc;
data _null_;
    /* infile cards */
    input @;
    _infile_=compress(resolve(_infile_));
    input a $ 1-3 b $ 4-6 c $ 7-9;
    put (_all_) (=);
cards4;
&x defghi
iiii
```

Q: What do we get for a,b,c?

A:

a=abc b=def c=ghi

NOTE: DATA statement used (Total process time):

Q: What happened here?

A: The RESOLVE function substituted "abc" for "&x" on the _infile_ buffer and wrote it back out. The compress function removed the blank.

Where Are You?

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

NOTE: There were 19 observations read from the data set SASHELP.CLASS.

NOTE: The data set WORK.CLASS has 19 observations and 5 variables.

A NULL WHERE CLAUSE WORKS!

```
data class;  
    set sashelp.class;  
    where(0);  
run;
```

NOTE: There were 0 observations read from the data set SASHELP.CLASS.

WHERE 0 /* an obviously FALSE where clause */ ;

NOTE: The data set WORK.CLASS has 0 observations and 5 variables.

Where Are You?

```
data class;  
    set sashelp.class;  
    where('0');  
run;
```

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
WHERE '0';

NOTE: The data set WORK.CLASS has 19 observations and 5 variables.

```
data class;  
    set sashelp.class;  
    if '0';  
run;
```

NOTE: Character values have been converted to numeric values at the places
given by: (Line):(Column). 33:9

NOTE: There were 19 observations read from the data set SASHELP.CLASS.

NOTE: The data set WORK.CLASS has 0 observations and 5 variables.

If you use the char variable by itself in a WHERE clause, the WHERE clause will select observations where the value of the character variable is not blank.

Where Are You?

```
data class;  
    set sashelp.class;  
    where not (sex in('M') and age gt 10);  
run;
```

(And by DeMorgan's Theorem)

NOTE: There were 9 observations read from the data set SASHELP.CLASS.

```
WHERE (sex not = 'M') or (age<=10);
```

NOTE: The data set WORK.CLASS has 9 observations and 5 variables.

```
data class;  
    set sashelp.class;  
    where not (sex in('T') and age gt 50) or age gt 10;  
run;
```

NOTE: There were 19 observations read from the data set SASHELP.CLASS.

```
WHERE 1 /* an obviously TRUE where clause */ ;
```

NOTE: The data set WORK.CLASS has 19 observations and 5 variables.

A Few Brief Comments

Q: Which of the following are SAS Comments?

1. `/* I am a comment */`
2. `* I am a comment;`
3. `%* I am a comment;`
4. `comment I am a comment;`

A: All of them!

1. is a block-style comment,
2. and 4. are statement-style (line) comments, and
- 3 is a macro comment.

A Few Brief Comments

Q: What happens when you use data-step line comments inside a macro?

```
%macro _testit(x);  
%* This is a comment;  
* %let x = 2;  
%put x = &x;  
%mend _testit;  
%_testit(3);
```

A: And the answer is:

x = 2

The Macro Processor executed the %let statement you thought you had commented.

A Few Brief Comments

Q: Why do I have two quotes inside these macro comments?

```
%macro testvar(dsname,varname);  
%* Macro Function testvar checks if varname exists -- RJK;  
%* Returns -1 if it can't open the file;  
%* Returns 0 if doesn't exist, gt 0 (varnum order) if it does;  
%let dsid=%sysfunc(open(%str(&dsname)));  
%if &dsid le 0  
%then %do;  
    -1  
    %put %sysfunc(sysmsg()) (in macro %nrstr(%testvar));  
%end;  
%else %do;  
    %sysfunc(varnum(&dsid,&varname))  
    %let rc=%sysfunc(close(&dsid));  
%end;  
%mend testvar;
```

A: It's a Quirk!

A Few Brief Comments

Q: What happens when you have a comment on the right-hand side of your %let statement?

```
%let string = %str(/*Landings are mandatory,*/ Takeoffs are optional);  
%put string = &string;
```

A: string = Takeoffs are optional

Under the MacroScope

```
%macro _a();  
data _null_;  
    set sashelp.class(obs=1);  
    call symput('name',trim(name));  
run;  
%mend _a;  
%_a;
```

```
%put name is: &name;
```

```
name is: Alfred    <<=SASLOG
```

OK, now delete the macro variable and add a parameter...

```
%symdel name;
```

Under the MacroScope

```
%macro _b(gender);  
data _null_;  
    set sashelp.class(obs=1);  
    call symput('name',trim(name));  
    where sex = "%upcase(&gender)";  
run;  
%put _user_;  
%mend _b;  
%_b(F);
```

```
NOTE: There were 1 observations read from the data set SASHELP.CLASS.  
      WHERE sex='F';
```

```
NOTE: DATA statement used (Total process time):
```

```
%put name is: &name;
```

```
WARNING: Apparent symbolic reference NAME not resolved.
```

```
name is: &name
```

This was working fine until I added the parameter.

Q: Where did the macro variable &name go?

Under the MacroScope

A: Inserting a `%put _user_;` in the macro reveals:

```
_B NAME Alice  
_B GENDER F
```

CALL SYMPUT does not open up a symbol table to place a macro variable. It puts the macro variable in the most local non-empty symbol table. A local table exists because of the parameter 'gender'.

Two Solutions: (1) `%global name;` in the macro.
 (2) `%let name=;` in open code.

Under the MacroScope

```
%macro _c();  
proc sql;  
    create table class as  
    select * from sashelp.class;  
quit;  
data _null_;  
    set work.class(obs=1);  
    call symput('name',trim(name));  
run;  
%mend _c;  
%_c  
%put name is: &name;  
WARNING: Apparent symbolic reference NAME not resolved.  
name is: &name
```

Q: Now What Happened? Any Guesses?

Under the MacroScope

A: Inserting a `%put _user_;` in the macro reveals:

```
_C SQLOBS 19  
_C SQLLOOPS 29  
_C NAME Alfred  
_C SQLXOBS 0  
_C SQLRC 0
```

Ever since SAS V8.x, the macro variables created by Proc SQL (SQLOBS, SQLLOOPS, SQLXOBS, SQLXOBS, and SQLXMSG) are placed in the local symbol table. Since it's a non-empty table, macro variable `&name` is also written there.

SAS Usage Note 4736 Workaround -- Put

```
%global sqlxrc sqlxmsg sqlloops sqlobs sqlxobs;
```

In your `autoexec.sas` file.

Under the MacroScope

```
/* Timing is Everything... */
options validvarname=V7;
data _null_;
    call execute('options validvarname=upcase; run;');
    call execute("%put <compile-time>;
    %put sysfunc(getoption(validvarname));");
<compile-time:>
V7
    call execute('%put <run-time>;
    %put %sysfunc(getoption(validvarname));');
    call execute('%nrstr(%put <call execute>;
    %put %sysfunc(getoption(validvarname));)');
run;
<run-time:>
V7
NOTE: DATA statement used (Total process time):
NOTE: CALL EXECUTE generated line.
1  + options validvarname=upcase; run;
2  + %put <call execute:>; %put %sysfunc(getoption(validvarname));
<call execute:>
UPCASE
```

Under the MacroScope

Inserting a macro variable between single quotes:

```
%let qtable_name = %unquote(%str(%' &table_name%'));
```

Inserting single quotes around a macro variable list:

```
%let LBTEST = ALP ALT GRANC AST CREAT HB PLAT TBILI WBC;  
%let qlbtest =  
%str(%')%qsysfunc(tranwrd(%sysfunc(compbl(&lbtest)),  
%str( ),%str(%' %')))%str(%');  
%put qlbtest = &qlbtest;  
qlbtest = 'ALP' 'ALT' 'GRANC' 'AST' 'CREAT' 'HB' 'PLAT' 'TBILI' 'WBC'
```

NOTE That unmatched characters such as ' " () % and & need to be "escaped" with a "%" when used with %str, %nrstr.

Missing Something?

- There are 28 possible missing values, from "." (lowest), ".a" ... ".z" (highest). Comparison operators treat them as negative infinity.
- Testing for only "." could result in a logic error, especially if you may have multiple missing values.
- Using [if | where] <var> le .z – covers all possible missing values.
- If <var>; means "if not missing or zero"; but zero = "01Jan1960"d -- Be careful when testing dates.
- Using [if | where] **missing**(<var>) works correctly, for both numeric and character variables.
- Instead of:
If **missing**(<var_a>) and **missing**(<var_b>) ...
you can also use: if **nmiss**(<var_a>, <var_b>) = 2 ...

Missing Something?

```
data _null_;  
  a=1; b=2; c=.; d=4; e=5; f=6; g=.z;;  
  min  = min(of a--g);  
  sml  = smallest(2,of a--g);  
  med  = median(of a--g);  
  mean = mean(of a--g);  
  lrg  = largest(2,of a--g);  
  max  = max(of a--g);  
  put min= sml= mean= med= lrg= max=;  
run;
```

min=1 sml=2 mean=3.6 med=4 lrg=5 max=6

bold type = new with SAS 9.1

NOTE: Missing values are by default excluded from functions and procedures.

Missing Something?

```
data _null_;  
  do x=-1, 0, 1, 2, 3, 4, .z;  
    put x= x= hex16.;  
  end;  
run;
```

```
x=-1 x=BFF0000000000000  
x=0 x=0000000000000000  
x=1 x=3FF0000000000000  
x=2 x=4000000000000000  
x=3 x=4008000000000000  
x=4 x=4010000000000000  
x=Z x=Z
```

NOTE: DATA statement used (Total process time):

Q: What's Quirky about this?

A: Missing values do not display in hex!

Missing Something?

```
data a;  
    input chr_date $8.;  
    cards4;  
04-05-02  
UNK  
iiii  
data b;  
    set a(where=(input(chr_date, yymmdd8.) is not missing));  
run;  
ERROR: INPUT function reported 'ERROR: Invalid date value' while processing  
    WHERE clause.  
NOTE: There were 1 observations read from the data set WORK.A.  
    WHERE INPUT(chr_date, YMMDD8.) is not null;  
NOTE: The data set WORK.B has 1 observations and 1 variables.
```

Q: How do we avoid getting tripped up on this error?

Missing Something?

```
data a;  
    input chr_date $8.;  
    cards4;  
04-05-02  
UNK  
iiii  
data b;  
    set a(where=(input(chr_date,? yymmdd8.) is not missing));  
run;
```

Error Message Suppressed

```
NOTE: There were 1 observations read from the data set WORK.A.  
      WHERE INPUT(chr_date, YMMDD8., '?') is not null;  
NOTE: The data set WORK.B has 1 observations and 1 variables.
```

A: Use a Question mark modifier to suppress the error message.

Stringing You Along

```
data _null_;  
    string='';  
    dat_len=length(string);  
    mac_len=%eval(%length(%str()));  
    put (dat_len mac_len) (=);  
run;
```

Q: What's the length of a null string?

A: It depends who you ask!

```
dat_len=1 mac_len=0
```

```
NOTE: DATA statement used (Total process time):
```

The data step compiler says it's 1. The macro processor says it's 0.

NOTE: Using (length(trimn(x))) will report a zero length in a data step.

Stringing You Along

Most character functions (in 9.1) return as the length of the target string the length of the first argument. These include: LEFT, RIGHT, COMPBL, COMPRESS, STRIP, DEQUOTE, SUBSTR, INPUTC, PUTC, TRANSLATE, LOWCASE, UPCASE, TRIM, TRIMN, and REVERSE -- But Not SCAN and TRANWRD!

```
data strings;  
  length string $300;  
  string=repeat(' zero one two tree fower fife six seven eight niner',4);  
  ls=length(string);  
  x=scan(string,-1,' ');          lx=vlength(x);  
  y=tranwrđ(string,'two','too');  ly=vlength(y);  
  put (_numeric_) (=);
```

run;

```
INFO: Character variables have defaulted to a length of 200 at the places given  
by: (Line):(Column). Truncation may result.
```

```
90:6      x
```

```
91:6      y
```

```
ls=255 lx=200 ly=200
```

NOTE: The data set WORK.STRINGS has 1 observations and 6 variables.

Stringing You Along

```
proc format ;  
    value $bug  
        '0000' = 'This might be a quirk'  
        other = 'or maybe a feature' ;  
run;
```

```
data test1;  
    attrib text length=$4 format=$bug.;  
    text = '0000';  
    vlt=vlength(text); put vlt=;
```

```
run;
```

```
vlt=4
```

NOTE: The data set WORK.TEST1 has 1 observations and 2 variables.

Stringing You Along

```
data test2 ;  
  attrib text format=$bug. length=$4;  
  text = '0000';  
  vlt=vlength(text); put vlt=;  
run;
```

WARNING: Length of character variable text has already been set.

vlt=21

NOTE: The data set WORK.TEST2 has 1 observations and 2 variables.

Q: What happened?

A: The Data Step compiler took as the length of text the default length of \$bug format.

Stringing You Along

```
data _null_;
    retain string "'I am Night--Color me Black', 'Twilight Zone', 'Rod
Serling'";
    title=dequote(string);
    string=substr(string,index(string,',')+2);
    tv_show=dequote(string);
    string=substr(string,index(string,',')+2);
    author=dequote(string);
    put (title tv_show author) (=);
run;
title=I am Night--Color me Black tv_show=Twilight Zone author=Rod Serling
NOTE: DATA statement used (Total process time):
```

The DEQUOTE function removes matching quotation marks from a character string that begins with an individual quotation mark and deletes everything that is to the right of the closing quotation mark. -- It just looks quirky!

How Much Was That Again?

```
/* Using an informat with decimals */
```

```
data receipts;  
    input amount 7.2;  
    cards4;
```

```
99.25
```

```
45.25
```

```
5.25
```

```
40
```

```
10.25
```

```
iiii
```

NOTE: The data set WORK.RECEIPTS has 5 observations and 1 variables.

```
proc summary nway data=receipts print nway sum n; var amount; run;
```

Q: What did we end up with?

Sum	N	Instead of 200, we ended up with 160.40. When using a W.D numeric informat, SAS divides the input data by 10**d unless the data contains a decimal point. Gotcha!
160.4000000	5	

What is the Meaning of This?

```
data _null_;  
  a=2; b=5;  
  c1 = a ne b;  
  c2 = a <> b;  
  c3 = a >< b;  
  put c1= c2= c3=;  
run;
```

Q: So what do we get for C1, C2, C3?

A: a=2 b=5 c1=1 c2=5 c3=2

NOTE: The "<>" operator is interpreted as "MAX".

NOTE: DATA statement used (Total process time):

Coding tip: Make it clear, don't use special chars.

Use FORTRAN codes (EQ, NE, GT, LT, etc.) and Functions.

Proc Quirkyness

```
data class;  
    set sashelp.class(sortedby=name);  
    where age is missing;  
run;
```

NOTE: There were 0 observations read from the data set SASHELP.CLASS.
WHERE age is null;

NOTE: The data set WORK.CLASS has 0 observations and 5 variables.

```
proc transpose data=class out=class(drop=_name_);  
by _all_;  
var height weight;  
run;
```

Q: And the number of observations in work.class?

A:

NOTE: There were 2 observations read from the data set WORK.CLASS.

NOTE: The data set WORK.CLASS has 2 observations and 7 variables.

Danger, Merge Ahead

There have been many presentations on Merge traps by notables such as Bob Virgile and Ian Whitlock. And Malachy Foley seems to have spent half his career researching them! So I won't get into any details here.

But Here is a tip: Keep It Short and Simple. And do your If/Then/Else and Where processing somewhere else. You are far less likely to get in trouble this way:

```
data <merged>;  
  merge <aa_data in=ina> <bb_data in=inb>;  
  by <by_vars>;  
  if ina /* and/or inb */;  
run;
```

You Know You Are a Pilot When

You've read the entire owner's manual for your car.

You get in your car, put on your lap belt, and your foot on the brake, roll down the window and yell 'clear!' before starting the engine.

You give yourself a 'take-off safety brief' as you leave your driveway.

Driving your car off the centerline feels unnatural to you.

You pull back on the steering wheel when your car hits 60 MPH.

Searching for "carb heat" on a cold morning, you accidentally yank a knob off the car radio.

You look for emergency landing spots while driving along in your car.

You reach for the hand brake to 'add flaps' when slowing down.

When you visit a new city by car you drive out to the GA airport, "Just to look around".

You can name all types of clouds when you look up at the sky.

You watch EVERY SINGLE plane that flies overhead.

You think flying in the sky is safer than driving on the road.



For More Information

SAS Software Support:

<http://support.sas.com/>

BASE SAS Documentation (9.2, 9.1 and even 8.2)

<http://support.sas.com/documentation/onlinedoc/base/index.html>

NorthEast SAS Users Group

<http://www.nesug.org/>

SAS-L Group

<http://groups.google.com/group/comp.soft-sys.sas/topics>

SAS Community Website:

http://www.sascommunity.org/wiki/Main_Page

Largest Searchable Collection of SAS User Presentations I Know of:

<http://www.lexjansen.com/>

About the Speaker

Rob Krajcik
Principal Analyst II

Bristol-Myers Squibb
5 Research PKWY
Wallingford, CT 06492-7660

(203) 677-6125 (phone)
(203) 677-6197 (fax)

robert.krajcik@bms.com

Introduction

NO, Sorry Try Again

[GoBack](#)

Introduction

The answer I got from SAS Technical Support:

"The developer said it's his speculation that even though one could expect up to 15 digits, if you wanted a 'put _all_' output (which uses BEST format) that you'd be able to see more data in fewer lines. Remember in those days there were punch cards and printed output, and it would conserve the paper."

[NextSlide](#)

SAS History

NO, Sorry Try Again

[GoBack](#)

SAS History

Given that IBM had the largest share of the mainframe market at the time, and was licensing its mainframes for next to nothing to universities, it made perfect sense for SAS to form a partnership with IBM.

[NextSlide](#)

SAS History

NO, Sorry Try Again

[GoBack](#)

SAS History

One of SAS's strengths was analyzing experiments with missing data, but the data step language was the one feature that made SAS stand out head and shoulders over its competition!

Anthony J Barr was the architect of the 1976 version of the Statistical Analysis System. The 1976 version is a comprehensive system for statistics, data management and report writing. According to his website, www.barrsystems.com, he designed and implemented all of the programming language, data management, report writing, and systems areas.

[NextSlide](#)

SAS History

NO, Sorry Try Again

[GoBack](#)

SAS History

PL/I (Programming Language 1) was a large, complex block-structured language invented by IBM, and first released in 1964 in conjunction with the influential System/360 line of computers.

PL/I was an attempt to compile the best features of Algol (program structure, semantics), FORTRAN (calculations), and COBOL (data structuring, I/O) into one new, all-purpose language. The result was a very complex language, but one that did serve most programming purposes quite well.

[NextSlide](#)

SAS History

Unlike PL/I, the Data Step section of a SAS program, assumes a default file structure, and automates the process of identifying files to the operating system, opening the input file, reading the next record, opening the output file, writing the next record, and closing the files.

This allows the user/programmer to concentrate on the details of working with the data within each record, in effect working almost entirely within an implicit data loop that runs for each record.

SAS has features that PL/I does not, such as: formats, automatic variables, and an implied data loop with an implied output statement.

[NextSlide](#)

SAS History

NO, Sorry Try Again

[GoBack](#)

SAS History

Doug Cockrell implemented the first version of the macro processor for SAS V82. The old style macros were quickly replaced, for good reason!

```
macro lt ge %  
data _null_;  
    a=5;  
    if a lt 10  
    then put 'a is less than 10.';  
    else put 'a is greater than 10.';  
run; /* from Jerry Le Breton */  
a is greater than 10.  
NOTE: DATA statement used (Total process time):
```

[NextSlide](#)

SAS History

NO, Sorry Try Again

[GoBack](#)

SAS History

YES! In fact this was the first portable release of SAS.

"Our [SAS's] acquisition of Lattice Inc. [in 1987] allowed us to gain the C expertise and compiler experience to successfully create a C implementation for PC-DOS and to move forward on the first truly portable version across all platforms. 6.06 was the first version that used common code for IBM mainframe (MVS and CMS), minicomputers (VAX, DG, and Prime), PC-DOS, and a number of Unix boxes (HP-UX, SunOS, and AIX"

-- Rick Langston, personal communication.

[NextSlide](#)

SAS History

NO, Sorry Try Again

[GoBack](#)

SAS History

YES, You got it! (6.09E was MVS, while 6.11 was Windows/Unix)
In fact, now you could write ENTIRE programs using ONLY macro code.

```
%macro manifest;  
  /*-----open rptfile-----*/  
  %let fida = %sysfunc(fopen(rptfile,a));  
  %if &fida = 0  
  %then %do;  
    %put %sysfunc(sysmsg());  
    %goto end_rept;  
  %end;  
  /*-----report header-----*/  
  %let strng = %unquote(%str(Requested sections that will be processed:));  
  %let rc    = %sysfunc(fput(&fida,%str(-)&strng));  
  %let rc    = %qsysfunc(fwrite(&fida));  
  /*-----output list of requested sections-----*/  
  %let dsid = %sysfunc(open(work.req_sect));  
  ...
```

[NextSlide](#)

SAS History

NO, Sorry Try Again

[GoBack](#)

SAS History

YES, You got it!

SAS V7 was actually a precursor to V8.x

SAS Institute recommended that sites wait until V8 before deploying the new software.

[NextSlide](#)

SAS History

NO, Sorry Try Again

[GoBack](#)

SAS History

YES, You got it!

OK Enough of this.

Next we're going to look at what's NOT a quirk...

[NextSlide](#)