

CAUTIONARY TALES FOR SAS PROGRAMMERS (or, Coding Pitfalls for the Unwary)

Jerry Le Breton

ABSTRACT

The ease of getting started with Base SAS® programming belies the breadth and depth of the SAS language. It can also lead users into trouble. Since users can generate results by writing just a minimal amount of code, and leaving the SAS System to 'fill in the blanks', they often don't realise or understand what SAS is doing for them behind the scenes. Its often not what they expect. Experienced users are certainly not immune either. Indeed, the experienced users often create more serious errors because of their greater, and sometimes misplaced, confidence.

Being aware of the most dangerous features will, of course, avoid potential problems, but so too will good programming practice - long before all the manuals have been consigned to memory.

This paper illustrates a few classic traps, and how to avoid them. It also demonstrates that the know-alls, don't!

INTRODUCTION

SAS users are often isolated from mainstream IT staff, and other SAS programmers too, so Alice felt lucky, on the whole, to be sharing a bay, with such a long time user like Alex. Alice had been struggling with her current task, and suddenly spun around on her chair in frustration. She came to rest, facing Alex's, temporarily abandoned, desk and surveyed the forest of plants that covered and surrounded it. She wondered how they did so well, so far from any natural light.

Through the undergrowth Alice could just see Alex's bookshelf and his collection of old Version 5 SAS manuals. She shook her head and turned back to her own partition's single bookshelf, bowing under the weight of her proud, new acquisition: an almost complete set of Version 6 manuals.

Everyone thought she was crazy, spending her own money on manuals the Company should buy. But with the Company downsizing and constantly bemoaning the cost of their SAS license, they had refused to buy any more manuals, so what was she supposed to do? She'd been programming with SAS for just 4 months and she couldn't keep asking Alex for help all the time - even if he was

always available; and Alex's dog-eared, Version 5 manuals really were out of date, even though they were still fine for a lot of things. At least her new manuals were tax deductible.

The challenge now, she found, was finding the time to absorb her new treasure trove.

A SHORT PROBLEM

"Alex, you're back! Got a minute?"

Carefully carrying a large jug of water, Alex grunted ambiguously and proceeded to start measuring out precise amounts of water for each plant.

"You know that job I was having trouble with, to identify dodgy customers? Well, since you weren't here, I rang SAS Institute Tech. Support and they suggested I cut the code down to the minimum and fax it off to them. Well, I did that even though the code looks pretty silly now, but I haven't heard back from them yet. Do you want to take a look? You'll probably see the problem straight away."

Alex took the bait. "Sure," and stretched out an arm to accept the proffered listing:

```
*****;
* DEFINE VALID VALUES *;
*****;
PROC FORMAT;
  VALUE $VALID
    '123','133'-'139','203'-'229'='1';
RUN;

*****;
* PICK VALID DATA FROM THE INPUT
* STREAM;
*****;
DATA VALID (KEEP=IDNO);

  INFILE CARDS MISOVER EOF=LASTREC;
  INPUT IDNO $ ;
  /* COUNT RECS WITH NO ID */
  IF LENGTH(IDNO) = 0 THEN
    NO_VALS + 1;

  /* OUTPUT ONLY VALID IDS */
  IF PUT(IDNO,$VALID.)='1';
RETURN;

LASTREC:
  PUT NO_VALS=;
RETURN;
```

CARDS;

123

234

111

136

144

;

RUN;

PROC PRINT;

RUN;

"So what's it doing then?" Alex demanded, after failing to spot anything obviously wrong.

"Well look. I've got it on my PC." Alice turned back to her desk to submit the code for the umpteenth time, while Alex came to peer over her shoulder. "It should print out 123 and 136 and a value 1 for NO_VALS since there is one blank id, but look", she said, "would you have expected this?"

OBS	IDNO
1	123
2	136
3	100
4	144
5	

"And this?" Alice added, switching to the Log.

NO_VALS=0

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

NOTE: The DATA statement used 0.11 seconds.

Alex frowned. No, he hadn't expected that. So he attacked.

"Why are you using the EOF= option on the input statement?" he asked, pointing the almost empty jug towards the screen. In six years of SAS programming he'd never used it before.

"I just thought I'd try it instead of the END= option. It saves having to code an IF statement to test for the END= variable. And it might be more efficient having SAS test for the end of file, instead of my IF statement. As you'd know, it just goes to the LASTREC label when it tries reading past the last record."

"Yes," said Alex lapsing back into thought, assimilating this new information.

"Of course," he announced, "you should always use an explicit OUTPUT statement. Instead of the subsetting IF, use an OUTPUT statement if the PUT function returns a '1'," he directed.

"OK" said Alice, considering the effect this would have. "Here goes." she said as she submitted the code again, with the IF test now reading:

```
IF PUT(IDNO,$VALID.)='1' then
  output;
```

"Well, that's got rid of that blank record" Alex crowed, feeling decidedly more confident now:

OBS	IDNO
1	123
2	136
3	100
4	144

"Hmm," said Alice, "so even though the last iteration of the data step didn't find any more data to read, and jumped to the LASTREC label, it still wrote out another record to the VALID data set."

"Yep, because of the implied OUTPUT statement SAS puts in before every explicit, or implicit, RETURN statement."

"And by using an explicit OUTPUT we stopped SAS inserting any other implied ones."

"OK. Now, what have you done to get these extra records?" Alex pressed on, choosing to ignore the mystery of NO_VALS being zero for the time being.

"There's not exactly much to go wrong", pleaded Alice. "How can it possibly find 100 in the \$VALID format?"

Good question, thought Alex, but he'd spotted a point to criticise and it might make a difference.

"Why haven't you got an OTHER in the Format? You should always have an OTHER." Alex asserted.

"No particular reason," Alice replied "it just didn't seem necessary." as she idly added an OTHER to the Format anyway. "The full three digit value will get returned if its not in the valid list and that can't be equal to '1'."

As she finished speaking and hit F3 to submit the code again, she suddenly felt a nagging doubt about her last statement. The Output window popped up to leave her in doubt about her doubt.

OBS	IDNO
1	123
2	136

"There you go." trumpeted Alex. "I'd better get back to my work now, Alice" Alex hoped to escape while he was ahead, but Alice wasn't letting him off the hook yet.

"But why, Alex? Why didn't it work before?"

But before Alex could say anything, Alice came to his rescue: "I know! The IF statement isn't comparing 100 with 1, at all. Its comparing 1 with

1 because the 100 is being truncated to just the first character."

Alice paused momentarily, then said "I'll bet if we put a 3 on the end of the format, like this..."

```
IF PUT(IDNO,$VALID3.)='1' then
  output;
```

"...we'll get all 3 characters returned by the PUT function when the id isn't in the format. Hey, I'll just add another line to see this properly."

```
a = PUT(IDNO,$VALID3.);
put a;
```

"And I'll take the OTHER out of the format again... Now watch this," and Alice excitedly hit F3 yet again.

The Output window showed the two expected ids and the Log proved Alice's assertion.

```
A=1
A=
A=234
A=1
A=100
A=144
NO VALS=0
NOTE: The data set WORK.VALID has 2
observations and 1 variables.
NOTE: The DATA statement used 0.11
seconds.
```

"Still, its a lot simpler to just use an OTHER in your formats," insisted Alex. How did she know to do that, thought Alex. He'd never seen a length specified on a user defined format before.

For good measure, he added "And of course, if you'd used a 'Y', or something, instead of a '1' as the label, you wouldn't have had any trouble either."

"Hang on," Alice called to the Alex's retreating back, "there's still the problem of why NO_VALS is zero."

Alex reluctantly turned back again.

"I don't need a RETAIN statement for NO_VALS, do I?" Alice asked.

"No. That form of expression automatically RETAINS the variable's value. The..."

But Alex was saved by the phone. Alice answered it.

"Speaking.

"Well actually, the local SAS guru has come back and we've just about sorted it out now. Just looking at why NO_VALS is zero..."

"Yes..."

"Yes..."

"You're kidding!"

"But that's crazy. It's totally counter-intuitive.

"Well yes, I guess I'll have to just say IF IDNO=''; but I used the LENGTH function 'cause I'm still going to need to know how long the id is, for some other processing.

"Yes.

"OK... and thanks. Bye."

Alice turned round to Alex, with an incredulous expression and shook her head.

"Amazing. You were no doubt about to tell me, but even the guy in Tech Support hadn't realised it. Must've still had his L-plates on - like me." Alice had started flicking through the Language Reference, looking for the LENGTH function.

"Who'd have thought a LENGTH function would return a value of 1 for a null string? Yep, here it is," she said, finding the page.

"If the value of the argument is missing, LENGTH returns a value of 1." Alice quoted.

"I guess we can't complain its not documented but hey, how many people would bother to read the fine print of such a simple function," she added.

Still gazing aghast at the manual, Alice hadn't noticed the expressions crossing Alex's face. Firstly, there had been disbelief, closely followed by panic. Alex turned hurriedly back to his PC, still holding his water jug, but no longer concerned about the condition of his plants - there were a couple of his production programs he suddenly felt the urge to have another look at.

Summary

"Well," said Alice, more to herself than to Alex. "Its hard to believe there could be so much wrong in those few lines of code.

"I should have used an explicit OUTPUT statement.

"I should have used an OTHER range on my format.

"I should have chosen my format label more carefully.

"And I should make sure I *really* understand the statements I use in my programs, and not to assume anything. That's hard."

Alice turned round to Alex looking for agreement, but he wasn't listening.

A MANUAL EXERCISE

"What!" exclaimed Alice suddenly, to her PC. "Hey, Alex, come and have a look at this."

Alex sat up and tried to arrange his face into an expression of concentration. He'd known it was

going to be a mistake having those extra beers with lunch.

"Since I finished that job to identify those dodgy customers, yesterday, I've had a chance to have a look at the 6.07 Changes and Enhancements. There's heaps in here, you know," Alice said, patting her new P-222 Technical Report that was lying open on her desk.

"You must have used the SET statement with the KEY= option, you know, to do direct, indexed reads of a SAS data set. I didn't realise SAS could do this. So anyway, I've just tried it out, and look what I get," Alice continued, pointing to the Log window on her screen.

Alex climbed out of chair, and through the leaves of the largest Kentia Palm.

'Aah, could we start with the source in the Editor rather than the Log," he asked, stifling a yawn.

"Sure."

```
*****;
* CREATE INDEXED LOOKUP DATASET *;
*****;
DATA NAMES (INDEX=(ID));
  INPUT ID      $ 1-3
        NAME    $ 4-20;
CARDS;
  1FRED SMITH
  13ANNE BROWN
  24JOHN WILLIAMS
;

*****;
* READ DATA FOR EACH PERSON AND LOOK
* UP THEIR NAME *;
*****;
DATA SALES;
  INPUT ITEM    $CHAR3.
        ID      $CHAR3.
        QTY     3.;
  SET NAMES KEY=ID;
  IF _IORC_=%SYSRC(_DSENOM) THEN
    PUT 'REC NOT FOUND';
CARDS;
A01  1  2
A02  1  2
B01 32  1
A01 13  2
A02 13  3
B01 24  2
;

*****;
* SUMMARISE AND PRINT RESULTS *;
*****;
PROC SUMMARY MISSING NWAY;
  CLASS NAME;
  VAR  QTY;
  OUTPUT SUM=;

PROC PRINT;
RUN;
```

"Pretty simple, eh? Did you say you'd used it?"

"Mmm, not for while."

"So, the first Data step just creates a data set with ID as the index, and the second Data step reads some data which also has a variable called ID, the value of which is used in the SET statement to find the name.

"Then I just summarise by NAME and print. Don't worry about the ITEM variable. I was going to do something else with that later.

"OK?" and Alice hit F3.

"Fred should have a QTY of 4, Anne 5, and John should have 2," Alice said, waiting for the Output window to pop up.

"And I get this."

OBS	NAME	_TYPE_	_FREQ_	QTY
1		1	6	12

"And look at the Log."

```
REC NOT FOUND
RULE:-----+-----1-----+-----2-----+-----3---
34  A01  1  2
ITEM=A01 ID=1 QTY=2 NAME= _ERROR_=1
_IORC_=1230015 _N_=1
REC NOT FOUND
35  A02  1  2
ITEM=A02 ID=1 QTY=2 NAME= _ERROR_=1
_IORC_=1230015 _N_=2
REC NOT FOUND
36  B01 32  1
ITEM=B01 ID=32 QTY=1 NAME= _ERROR_=1
_IORC_=1230015 _N_=3
REC NOT FOUND
37  A01 13  2
ITEM=A01 ID=13 QTY=2 NAME= _ERROR_=1
_IORC_=1230015 _N_=4
REC NOT FOUND
38  A02 13  3
ITEM=A02 ID=13 QTY=3 NAME= _ERROR_=1
_IORC_=1230015 _N_=5
REC NOT FOUND
39  B01 24  2
ITEM=B01 ID=24 QTY=2 NAME= _ERROR_=1
_IORC_=1230015 _N_=6
NOTE: THE DATA SET WORK.SALES HAS 6
OBSERVATIONS AND 4 VARIABLES.
NOTE: THE DATA STATEMENT USED 5.76
SECONDS.
```

"It gave me all these errors but eventually, with the help of that really friendly %SYSRC macro, I found out what it was. Wish they'd give us a macro to simply describe what the error code means. It would be a lot easier.

"Anyway, it reckons it can't find the records on the NAMES data set. Any suggestions?

"Look, I'll print the two data sets," and Alice submitted a couple of PROC PRINTs.

OBS	ID	NAME
-----	----	------

```

1      1      FRED SMITH
2      13     ANNE BROWN
3      24     JOHN WILLIAMS

```

OBS	ITEM	ID	QTY	NAME
1	A01	1	2	
2	A02	1	2	
3	B01	32	1	
4	A01	13	2	
5	A02	13	3	
6	B01	24	2	

"Well, there's the problem," Alex announced quickly. "By using two different styles of INPUT statement, you've managed to left-justify ID in NAMES and right-justify it in SALES."

"Oh, yes," said Alice, switching back to the Editor. Alex continued: "You could use a \$3. instead of the \$char3. to get rid of the leading spaces, but its usually better to use just one style of INPUT statement. Easier to understand - and less likely to trip you up."

"I usually do," said Alice, as she changed the informat to a \$3. "I only did this because it was a little test program."

"Hey, this is more like it."

OBS	NAME	_TYPE_	_FREQ_	QTY
1	ANNE BROWN	1	2	5
2	FRED SMITH	1	3	5
3	JOHN WILLIAMS	1	1	2

"Except Fred should only have 4," Alex pointed out.

"That's right," said Alice switching back to the Log. "Oh, but this isn't!"

```

REC NOT FOUND
RULE:-----+-----1-----+-----2-----+-----3---
142  A02  1  2
ITEM=A02 ID=1 QTY=2 NAME=FRED SMITH
_ERROR_=1 _IORC_=1230015 _N_=2
REC NOT FOUND
143  B01 32  1
ITEM=B01 ID=32 QTY=1 NAME=FRED SMITH
_ERROR_=1 _IORC_=1230015 _N_=3
REC NOT FOUND
145  A02 13  3
ITEM=A02 ID=13 QTY=3 NAME=ANNE BROWN
_ERROR_=1 _IORC_=1230015 _N_=5
NOTE: THE DATA SET WORK.SALES HAS 6
OBSERVATIONS AND 4 VARIABLES.
NOTE: THE DATA STATEMENT USED 3.83
SECONDS.

```

"Mmm, it's saying it still can't find those records," mused Alex.

"Yes. It found the first one for Fred, but not the second. That's ridiculous. And it found John OK."

"Yes..." said Alex, frowning, trawling his memory. "Isn't there something else you need with the KEY= option?"

"No. Look," and Alice pointed to the description of the KEY= option under the SET statement in the Changes and Enhancements manual. "And there's the example."

Alex shook his head. "Well, I don't know. You might have to call Tech Support again."

"Excuse me. I've got to go. I'll probably be about half an hour," said Alex, and grabbed a folder from the piles on his desk and headed off towards the lifts.

Alice picked up the phone and punched the familiar number.

Ten minutes later Alice was beginning to wonder if buying the manuals were more trouble than they were worth. There it was, just like the guy in Tech Support had said, but how on earth, was anyone expected to know that in order to use a feature introduced in version 6.07, you had to refer to the Changes and Enhancements for release 6.08!

Just then Alex stomped back. "They could've phoned me," he said. "It's bad enough wasting time at these meetings, let alone wasting time going to meetings that've been cancelled!"

Alice let him calm down, and then said: "You were quite right about the KEY= option on the SET statement, you know. There is something else."

"I thought so," said Alex, tuning back into Alice's problem, and hauling himself out of his chair again.

"Look," Alice said, and pointed to the SET statement described, this time, in the Changes and Enhancements Technical Report for 6.08, P-242.

"That's right," Alex said. "The UNIQUE option. Have you tried it yet?"

"I was just about to. According to the guy on Tech Support, it's pretty well essential," Alice said as she changed the SET statement.

```
SET NAMES KEY=ID/UNIQUE;
```

"If the main data set has consecutive records with the same key," Alice explained, "it seems SAS assumes there'll be the same pattern of records in the lookup data set. So, the second time it looks for a record for Fred in NAMES, it starts looking *after* the record it found last time, expecting to find another record with the same key - only it finds Anne. Does that sound logical to you? No."

Well, they presumably felt the same, so we now have the UNIQUE option to force SAS to search the lookup data set's index from the beginning.

"OK, lets see what we get this time."

OBS	NAME	_TYPE_	_FREQ_	QTY
1	ANNE BROWN	1	2	5
2	FRED SMITH	1	3	5
3	JOHN WILLIAMS	1	1	2

"Well, that 's not a great improvement. Let's have a look at the Log."

```

REC NOT FOUND
RULE:-----1-----2-----3--
187 B01 32 1
ITEM=B01 ID=32 QTY=1 NAME=FRED SMITH
ERROR =1 IORC =1230015 N =3
NOTE: THE DATA SET WORK.SALES HAS 6
OBSERVATIONS AND 4 VARIABLES.
NOTE: THE DATA STATEMENT USED 6.15
SECONDS.

```

"OK, it's not finding ID 32, but why is NAME equal to 'Fred Smith' when it couldn't find a record in the lookup data set?" Alice asked. "Oh, but it explains why Fred is still getting a total of 5, of course. The QTY of one for this record, is being summed by the PROC SUMMARY with the other records for Fred, even though the ID is different."

"That's right," said Alex. "And of course, because no record is found by the SET statement, the value of NAME is still 'Fred Smith'."

There was silence for a moment as Alice thought about that.

"But the NAME variable, along with everything else, is re-initialised at the start of each iteration of the Data step," insisted Alice.

"Not so," said Alex, realising he had an opportunity to impress. "Do you know about the Program Data Vector?"

"Vaguely."

"It's the area of memory SAS uses to define all the variables in the Data step, and where the values are stored, before writing them to the output data set. Now," he continued importantly, "only variables defined by INPUT statements, LENGTH statements and assignment statements are reset at the top of the Data step. And you can use the RETAIN statement too, to override this behaviour."

"So, values for variables input with a SET statement will stay there until something, usually the same SET statement, replaces them."

"Of course, because the SET statement is usually at the top of the Data step, and there's usually only the one, people don't realise what's really going on - and it doesn't usually matter."

Seeing Alice's dubious expression, Alex pushed forward, towards Alice's keyboard. "May I?"

Alice waved him on and retreated to one side.

"No, no. Can I move this?" Alex pointed to Alice's sports bag, on the floor between his feet.

"Of course. Hey, I didn't tell you; I did the 10 kilometres in 39:40 at lunch time."

"Good grief, what ever for," said Alex, trying hard not to be distracted, as he pushed the bag out of the way with his foot.

"Look," he continued, "all I have to do to make this work is reset NAME each time, in case there's no match with the NAMES data set," Alex said, typing in his amendment.

```

DATA SALES;
INPUT ITEM $CHAR3.
      ID $3.
      QTY 3.;
NAME = '';
SET NAMES KEY=ID/UNIQUE;
IF IORC =%SYSRC( DSENM) THEN
PUT 'REC NOT FOUND';

```

"Now you'll see what I mean," and Alex pressed F3 with a flourish.

"That's interesting," said Alice, trying to contain a smile as the Output window opened.

OBS	NAME	_TYPE_	_FREQ_	QTY
1		1	1	1
2	A	1	2	5
3	F	1	2	4
4	J	1	1	2

With the wind momentarily taken out of Alex's sail, Alice had time to spot the problem and suggested: "The length of NAME is being set to one, I think Alex, because of the assignment statement."

"I know," Alex replied gruffly, and quickly typed in a LENGTH statement too,

```
LENGTH NAME $30;
```

and submitted the code again.

"There you are," Alex waved at the Output window.

OBS	NAME	_TYPE_	_FREQ_	QTY
1		1	1	1
2	ANNE BROWN	1	2	5
3	FRED SMITH	1	2	4
4	JOHN WILLIAMS	1	1	2

"Wonderful," exclaimed Alice, slumping back in her chair with relief.

Summary

"Well, I'm glad its Friday," Alice said, "I think I've learnt quite enough, in the last couple of days.

"Let's see, on top of yesterday's lessons, I've remembered its cruel to ask for help from people who've just had lunch down at the pub.

"I now have a better idea how the Program Data Vector works, and that not all variables are reset at the start of each iteration of the Data step.

"I've learnt to be more careful with my INPUT statements, and especially how I read character variables with leading spaces.

"Alex, you reminded me that letting SAS decide how long a variable should be can cause problems.

"And I've learnt *all* about the KEY= option on the SET statement."

Alice paused.

"Yesterday, I learnt to check the manual, just in case I didn't really know what I thought I knew. Today, I've learnt to read *all* the manuals, just in case there's something I *should* know but didn't know that I didn't know.

"Oh, I love SAS!"

SAS is a registered trademark of SAS Institute Inc. in the USA and in other countries. ® indicates USA registration.

Author Contact:

Jerry Le Breton

7 Way Street

O'Connor

ACT 2602

Australia

lebreton@spirit.com.au