Paper 063-2012

Deciphering PROC COMPARE Codes: The Use of the bAND Function

Joseph Hinson, Merck Sharp & Dohme Corp., Rahway, NJ
Margaret Coughlin, Merck Sharp & Dohme Corp., Rahway, NJ

ABSTRACT

The COMPARE procedure is very useful for validating SAS® data sets in clinical studies. One particularly useful feature is its SYSINFO system macro variable by which numerical codes are issued following the comparison of data sets. These codes represent up to 16 messages that describe the outcome of the comparisons. Thus, interpreting SYSINFO codes can provide a very concise PROC COMPARE report, especially when many pairs of data sets are involved. In this paper, we demonstrate a novel approach for decoding SYSINFO values using the SAS function bAND (bitwise logical AND). With this function, one can quickly determine which of the 16 PROC COMPARE output messages a SYSINFO code contains.

INTRODUCTION

An essential part of establishing the integrity of clinical trial data is the validation of the data sets used to generate the listings, tables, and graphs in Clinical Study Reports (CSR). To achieve that goal, the SAS® Compare procedure becomes a very convenient tool.

However, the standard PROC COMPARE reports are often overwhelming and cumbersome. Results of comparison can run into many pages and can prove challenging when one is handling dozens of folders containing hundreds of data sets.

Rather fortunately, PROC COMPARE also produces a compact reporting system by issuing return codes based on the outcome of the comparisons. These return codes are stored in the automatic system macro variable called SYSINFO. In this paper, we present a novel approach of interpreting SYSINFO codes using the SAS® bitwise function called "bAND" or "bitwise AND". This is possible because SYSINFO codes are binary in nature.

SYSINFO CODES

The SYSINFO return codes represent binary numbers.
PROC COMPARE looks for 16 possible differences between data sets (see Table 1).
These 16 differences can be viewed as 16 bit positions in a binary number of 16 digits:

16th	15th	14th	13th	12th	11th	10th	9th	8th	7th	6th	5th	4th	3rd	2nd	1st
0	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**1**

where **0** at a particular position means the difference is absent and **1** means the difference has been detected.

For example, **1** at bit position 12 means Difference#12 was found in the comparison:

16th	15th	14th	13th	12th	11th	10th	9th	8th	7th	6th	5th	4th	3rd	2nd	1st
0	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

where Difference#12 is: "Comparison data set has variable not in Base".

The binary code for **0000100000000000** is **2048** (decimal form)**.**
Thus when SYSINFO code is 2048, it means "Comparison data set has variable not in Base".
That is the basis for interpreting SYSINFO codes.

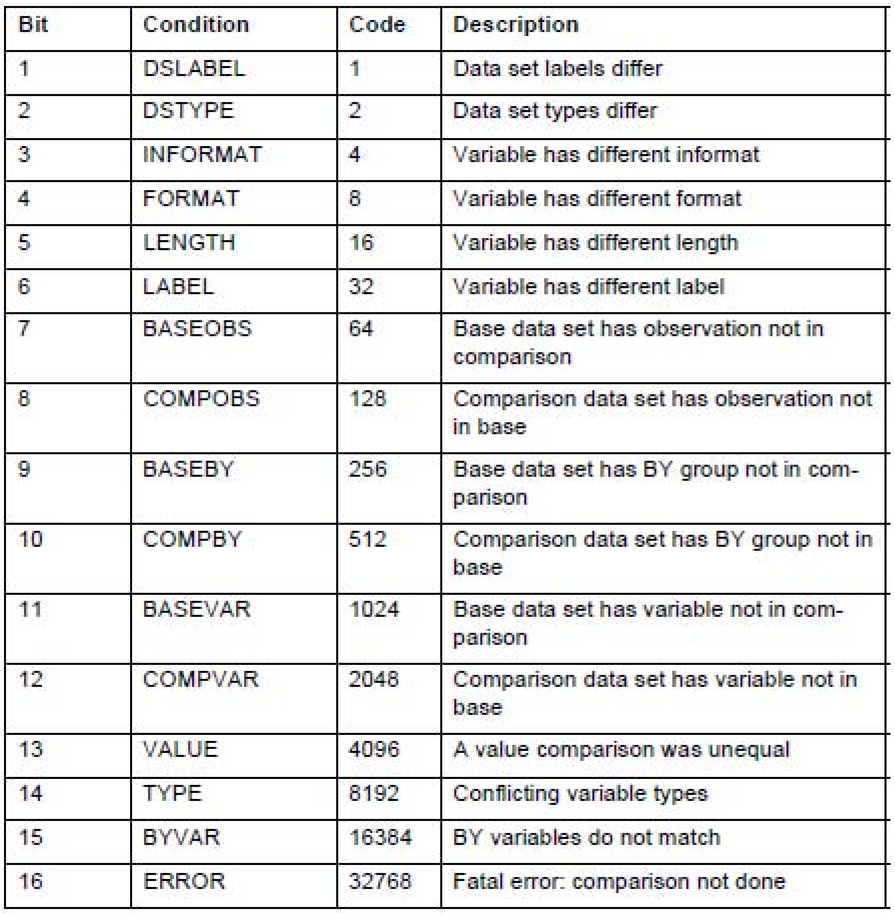
When many of the 16 differences are found in a single comparison, the binary codes are added.

Thus, SYSINFO=**14369** decodes to: "**0011100000100001**" in binary,
which can be interpreted as:

"Data set labels differ"	(1st bit position =1)
"Variable has different label"	(6th bit position =1)
"Comparison data set has variable not in Base"	(12th bit position =1)
"A value comparison was unequal"	(13th bit position =1)
"Conflicting variable types"	(14th bit position =1)

So a single decimal number provided by SYSINFO can hold several messages, making reporting very compact. But there is still the drudgery of obtaining the messages hidden in those codes.

Table 1: The Meaning of SYSINFO Codes



Bit	Condition	Code	Description
1	DSLABEL	1	Data set labels differ
2	DSTYPE	2	Data set types differ
3	INFORMAT	4	Variable has different informat
4	FORMAT	8	Variable has different format
5	LENGTH	16	Variable has different length
6	LABEL	32	Variable has different label
7	BASEOBS	64	Base data set has observation not in comparison
8	COMPOBS	128	Comparison data set has observation not in base
9	BASEBY	256	Base data set has BY group not in com-parison
10	COMPBY	512	Comparison data set has BY group not in base
11	BASEVAR	1024	Base data set has variable not in com-parison
12	COMPVAR	2048	Comparison data set has variable not in base
13	VALUE	4096	A value comparison was unequal
14	TYPE	8192	Conflicting variable types
15	BYVAR	16384	BY variables do not match
16	ERROR	32768	Fatal error: comparison not done

INTERPRETING SYSINFO CODES

There are many easy ways to interpret SYSINFO return codes.

(a) The easiest and simplest but inefficient way is to use just IF statements:

if sysinfo=0 then message="There is no difference between Base data set and Comparison";
if sysinfo=64 then message="Base data set has observation not in Comparison";
if sysinfo=32768 then message ="Fatal error: comparison not done";

(b) Using IF statements with Bit Masks:

This approach is quite efficient although it requires sixteen IF statements.
SYSINFO codes, being binary numbers, can be subject to "bit testing", the process of detecting 1's in particular positions in a binary number. A "bit mask" is a SAS® binary literal used for bit testing. The binary literal**, '………..1'b** , can be used as a bit mask in a bit test. A bit test is the comparison of the binary form of a number against a bit mask. The periods in a bit mask are neutral place holders and yield a TRUE result regardless of the value of the corresponding bit.
In other words, if a binary number is: 000100110100, and the bit mask is "………..1..", that bit-mask would detect just the "1" in the 3rd position from the right, ignoring all other 1's and 0's elsewhere. Bit masks therefore detect the presence of "1" in a particular position in a binary number.

To interpret a SYSINFO code, a bit mask is used as follows:

if sysinfo='1'b then message= "Data set labels differ";
if sysinfo='1.'b then message= "Data set types differ";
if sysinfo='1..'b then message= "Variable has different informat";
if sysinfo='1…'b then message= "Variable has different format";
if sysinfo='1….'b then message= "Variable has different length";
if sysinfo='1…..'b then message= "Variable has different label";
and so on.

THE BITWISE-AND (BAND) FUNCTION

SAS® provides six bitwise logical functions: bAND, bLSHIFT, bNOT, bOR, bRSHIFT, and bXOR.
The bAND function returns the bitwise logical AND of two arguments.
The syntax is: bAND (argument-1, argument-2), where the arguments can be numbers, variables, or expressions.

SYSINFO INTERPRETATION WITH BAND

The nice thing about bAND which is exploited for SYSINFO decoding is this:

if a binary code x contains another binary code y, then bAND (x, y) = y, else bAND (x, y) = 0.

For example, earlier we showed that the SYSINFO code 14369 contains 4 messages coded by 1, 32, 2048, and 4096 respectively.
Thus,
bAND (14369, 1) = 1
bAND (14369, 32) = 32
bAND (14369, 2048) = 2048
bAND (14369, 4096) = 4096

By contrast,
bAND (14369, 2) = 0
bAND (14369, 512) =0

because SYSINFO 14369 does not code for "Data set types differ" (code 2),
nor does it code for "Comparison data set has BY group not in Base" (code 512).

Thus the bAND function is pretty effective for unraveling the messages embedded in SYSINFO return codes.

DATA SET VALIDATION PROGRAM USING BAND

The use of the bAND function with SYSINFO return codes enables the efficient validation of a large set of SAS® tables. In fact, the validation program we have developed, "ValidateDatasets.sas", compares folders containing large numbers of data sets, producing a single table of report for each folder, by means of SYSINFO codes and their meanings.

The program is organized as follows:

[SECTION-A] INPUT DATA PREPARATION:
Step 01: Create macro variables for file pathnames
Step 02: Pull data set names from **dictionary.tables** into macro variables

[SECTION-B] PROC COMPARE PROCESSING
Step 03: Prepare a table (**syscodes**) to receive PROC COMPARE sysinfo codes
Step 04: Run PROC COMPARE to match OLD data sets with NEW data sets

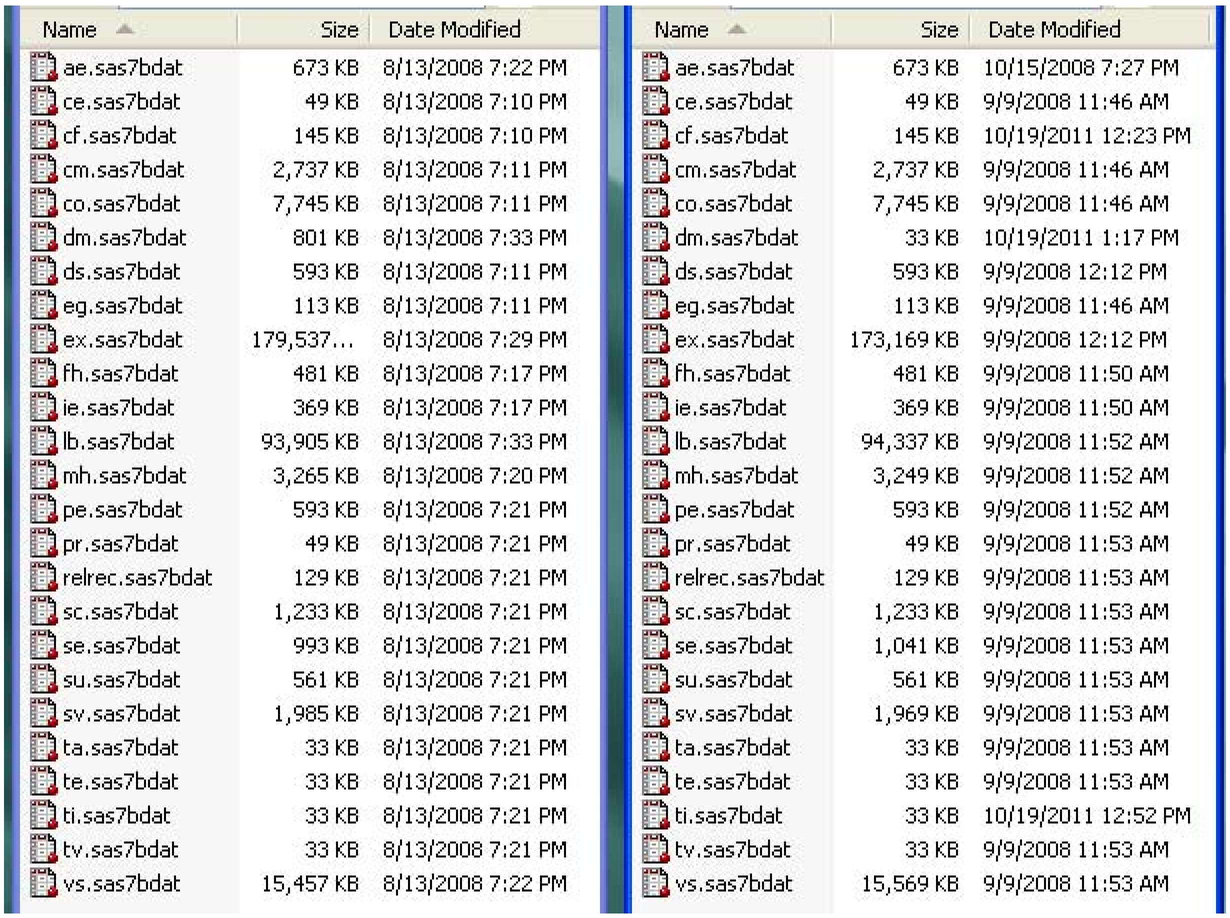
[SECTION-C] SYSINFO CODE PROCESSING
Step 05: Capture PROC COMPARE SYSINFO codes into macro variables
Step 06: Decode each SYSINFO code into a PROC COMPARE message
Step 07: Accumulate in SYSINFO table (**syscodes**) data set names,
SYSINFO codes, and decoded messages
Step 08: Call Macro

[SECTION-D] OUTPUT
Step 09: Print SYSINFO report from table (**syscodes**)

```
*-----------------------------------------------------------------------------------------------------------
  [SECTION-A]            I N P U T   D A T A   P R E P A R A T I O N
----------------------------------------------------------------------------------------------------------;

*----------------Step 01: Create macro variables for file pathnames------------------------------------------;
%let datapath=C:\Documents and Settings\hinsonj\Desktop\Publication;
%let study=123;
libname OLD&study "C:\Documents and Settings\hinsonj\Desktop\Publication\OLD123";
libname NEW&study "C:\Documents and Settings\hinsonj\Desktop\Publication\NEW123";

*---------- Step 02: Pull data set names from dictionary.tables into macro variables ------------------------;
proc sql noprint;
                 select memname into :OLDset1 - :OLDset&sysmaxlong
                         from dictionary.tables
                                 where indexw("OLD&study" , libname);

                 select memname into :NEWset1 - :NEWset&sysmaxlong
                         from dictionary.tables
                                 where indexw("NEW&study" , libname);
quit;
%let nsize=&sqlobs;

*-----------------------------------------------------------------------------------------------------------
  [SECTION-B]      P R O C     C O M P A R E     P R O C E S S I N G
----------------------------------------------------------------------------------------------------------;

%macro sys0proc0compare;

%*-------------Step 03:   Prepare a table to receive Proc Compare sysinfo codes -----------------------------;
        proc sql;
                create table syscodes
                (DATASET char(14), SYSINFO num, MEANING_OF_SYSINFO_CODES char(400));
        quit;

%*-------------Step 04:   Run Proc Compare to match OLD data sets with NEW data sets-------------------------;
        %do N=1 %to &nsize;

                proc compare
                        base=OLD&study..&&OLDset&N.
                        compare=NEW&study..&&NEWset&N.
                        ;
                run;

%*----------------------------------------------------------------------------------------------------------
```

```
  [SECTION-C]      S Y S I N F O     C O D E S     P R O C E S S I N G
------------------------------------------------------------------------------------------------------------;

%*-------------Step 05:   Capture Proc Compare SYSINFO codes into macro variables ---------------------------;
                 %let sicode=&sysinfo;
                 %let dsname=&&NEWset&N;

%*-------------Step 06:   Decode each SYSINFO code into a Proc Compare message ------------------------------;
                 data _null_;
                         length message $ 600 decoded $ 600  text $60;
                         array msg {17} $ 60 _temporary_ (
                                         " ",
                                         "Data set labels differ",
                                         "Data set types differ",
                                         "Variable has different informat",
                                         "Variable has different format",
                                         "Variable has different length",
                                         "Variable has different label",
                                         "OLD data set has observation not in NEW",
                                         "NEW data set has observation not in OLD",
                                         "OLD data set has BY group not in NEW",
                                         "NEW data set has BY group not in OLD",
                                         "OLD data set has variable not in NEW",
                                         "NEW data set has variable not in OLD",
                                         "A value comparison was unequal",
                                         "Conflicting variable types",
                                         "BY variables do not match",
                                         "Fatal error: comparison not done"
                                         );

                         testcode=&sicode;
                         testcode=&sicode;
                         if testcode=0 then decoded="NO DIFFERENCE BETWEEN OLD & NEW"; *<---(&SYSINFO=0
                                                                                    WHEN NO DIFFERENCE DETECTED);
                         else do;
                                 decoded=" ";  *<----------(VARIABLE 'DECODED' WILL STORE &SYSINFO MESSAGES);
                                 do k=1 to 16; *<----------(BECAUSE THERE ARE 16 POSSIBLE &SYSINFO MESSAGES);
                                   binval=2**(k-1);  *<--(CONVERT 0,1,2,3...16 TO BINARY 1,2,4,8,16,32,64..);
                                   match=band(binval, testcode);  *<--(DO BITWISE TESTING WITH BAND FUNCTION);
                                   key=sign(match)*k;  *<---(REVERT BINARIES TO REGULAR NUMBERS FOR ARRAY-
                                                                                                  INDEXING);
                                   text=msg(key+1); *<---(GET DECODED MESSAGE TEXT FROM ARRAY VALUE);
                                  decoded=catx(", ",decoded,text); *<---(CONCATENATE IF MORE THAN ONE &SYSINFO
                                                                                                    MESSAGE);
                                 end;
                              end;
                         call symputx("message",decoded);

                 run;


%*----Step 07:   Accumulate in SYSINFO table(syscodes) data set names, SYSINFO codes, and decoded messages --;
                 proc sql;
                         insert into syscodes values("&dsname.",&sicode.,"&message.");
                 quit;
         %end;

%mend sys0proc0compare;

*----Step 08:    Call Macro ----------------------------------------------------------------------------;
%sys0proc0compare;

*-----------------------------------------------------------------------------------------------------------
  [SECTION-D]                     O U T P U T
-----------------------------------------------------------------------------------------------------------;

*----Step 09:    Print SYSINFO report from table(syscodes)-----------------------------------------------;
ODS RTF FILE="C:\Documents and Settings\hinsonj\Desktop\Publication\syscodes.rtf";
ODS NOPROCTITLE;

proc print data=syscodes;
        title "Comparisons Between OLD and NEW data sets: A Sysinfo Report";
run;

ODS RTF CLOSE;
*_ _ _ _ _ _ _ _ _ _ _ _ _ END OF PROGRAM ValidateDatasets.sas _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ ;
```

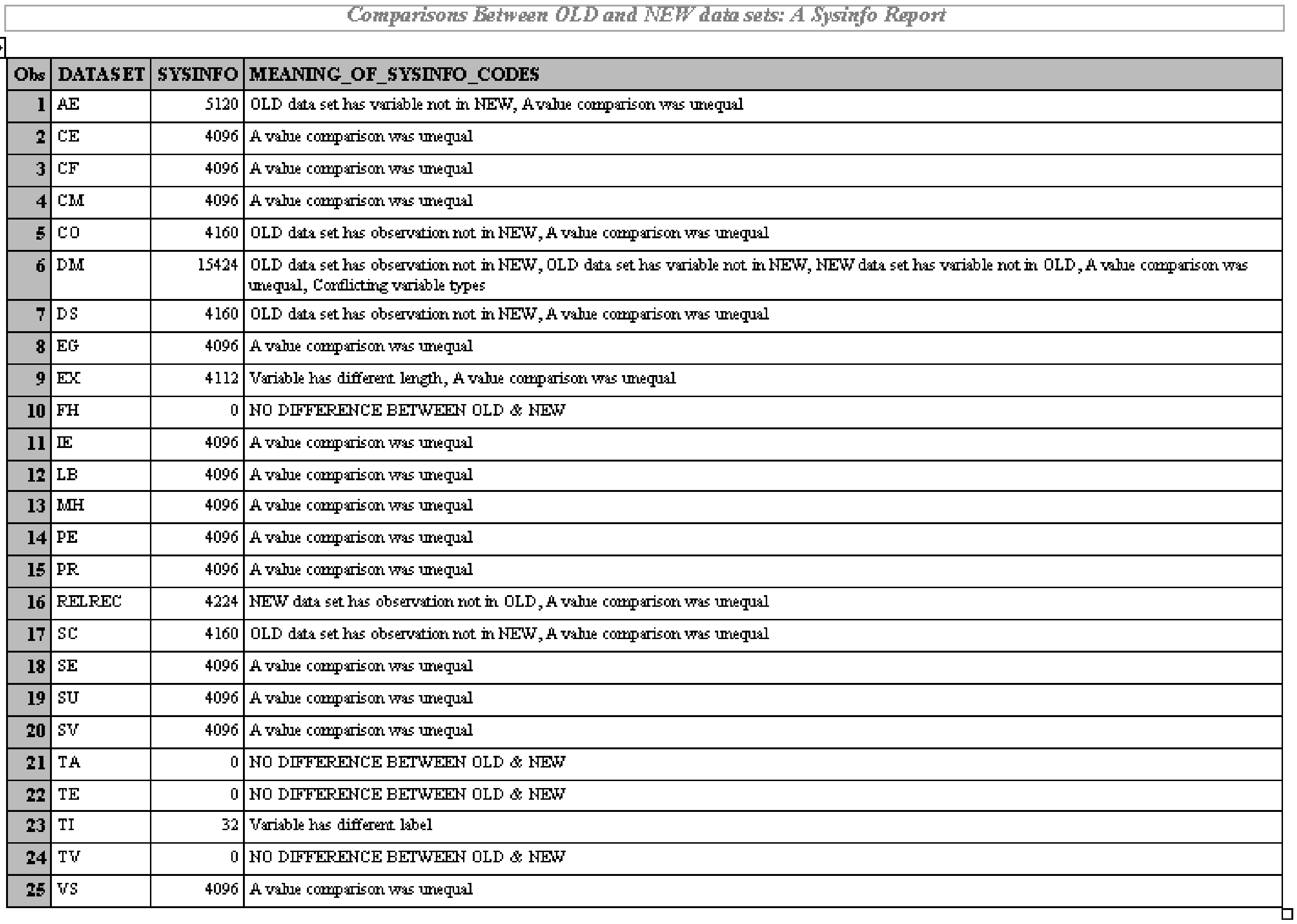
Table 2: Folders of Data Sets to Validate



Name	Size	Date Modified
ae.sas7bdat	673 KB	8/13/2008 7:22 PM
ce.sas7bdat	49 KB	8/13/2008 7:10 PM
cf.sas7bdat	145 KB	8/13/2008 7:10 PM
cm.sas7bdat	2,737 KB	8/13/2008 7:11 PM
co.sas7bdat	7,745 KB	8/13/2008 7:11 PM
dm.sas7bdat	801 KB	8/13/2008 7:33 PM
ds.sas7bdat	593 KB	8/13/2008 7:11 PM
eg.sas7bdat	113 KB	8/13/2008 7:11 PM
ex.sas7bdat	179,537…	8/13/2008 7:29 PM
fh.sas7bdat	481 KB	8/13/2008 7:17 PM
ie.sas7bdat	369 KB	8/13/2008 7:17 PM
lb.sas7bdat	93,905 KB	8/13/2008 7:33 PM
mh.sas7bdat	3,265 KB	8/13/2008 7:20 PM
pe.sas7bdat	593 KB	8/13/2008 7:21 PM
pr.sas7bdat	49 KB	8/13/2008 7:21 PM
relrec.sas7bdat	129 KB	8/13/2008 7:21 PM
sc.sas7bdat	1,233 KB	8/13/2008 7:21 PM
se.sas7bdat	993 KB	8/13/2008 7:21 PM
su.sas7bdat	561 KB	8/13/2008 7:21 PM
sv.sas7bdat	1,985 KB	8/13/2008 7:21 PM
ta.sas7bdat	33 KB	8/13/2008 7:21 PM
te.sas7bdat	33 KB	8/13/2008 7:21 PM
ti.sas7bdat	33 KB	8/13/2008 7:21 PM
tv.sas7bdat	33 KB	8/13/2008 7:21 PM
vs.sas7bdat	15,457 KB	8/13/2008 7:22 PM

Name	Size	Date Modified
ae.sas7bdat	673 KB	10/15/2008 7:27 PM
ce.sas7bdat	49 KB	9/9/2008 11:46 AM
cf.sas7bdat	145 KB	10/19/2011 12:23 PM
cm.sas7bdat	2,737 KB	9/9/2008 11:46 AM
co.sas7bdat	7,745 KB	9/9/2008 11:46 AM
dm.sas7bdat	33 KB	10/19/2011 1:17 PM
ds.sas7bdat	593 KB	9/9/2008 12:12 PM
eg.sas7bdat	113 KB	9/9/2008 11:46 AM
ex.sas7bdat	173,169 KB	9/9/2008 12:12 PM
fh.sas7bdat	481 KB	9/9/2008 11:50 AM
ie.sas7bdat	369 KB	9/9/2008 11:50 AM
lb.sas7bdat	94,337 KB	9/9/2008 11:52 AM
mh.sas7bdat	3,249 KB	9/9/2008 11:52 AM
pe.sas7bdat	593 KB	9/9/2008 11:52 AM
pr.sas7bdat	49 KB	9/9/2008 11:53 AM
relrec.sas7bdat	129 KB	9/9/2008 11:53 AM
sc.sas7bdat	1,233 KB	9/9/2008 11:53 AM
se.sas7bdat	1,041 KB	9/9/2008 11:53 AM
su.sas7bdat	561 KB	9/9/2008 11:53 AM
sv.sas7bdat	1,969 KB	9/9/2008 11:53 AM
ta.sas7bdat	33 KB	9/9/2008 11:53 AM
te.sas7bdat	33 KB	9/9/2008 11:53 AM
ti.sas7bdat	33 KB	10/19/2011 12:52 PM
tv.sas7bdat	33 KB	9/9/2008 11:53 AM
vs.sas7bdat	15,569 KB	9/9/2008 11:53 AM

Table 3: Proc Compare SYSINFO-Based Report



Comparisons Between OLD and NEW data sets: A Sysinfo Report

Obs	DATASET	SYSINFO	MEANING_OF_SYSINFO_CODES
1	AE	5120	OLD data set has variable not in NEW, A value comparison was unequal
2	CE	4096	A value comparison was unequal
3	CF	4096	A value comparison was unequal
4	CM	4096	A value comparison was unequal
5	CO	4160	OLD data set has observation not in NEW, A value comparison was unequal
6	DM	15424	OLD data set has observation not in NEW, OLD data set has variable not in NEW, NEW data set has variable not in OLD, A value comparison was unequal, Conflicting variable types
7	DS	4160	OLD data set has observation not in NEW, A value comparison was unequal
8	EG	4096	A value comparison was unequal
9	EX	4112	Variable has different length, A value comparison was unequal
10	FH	0	NO DIFFERENCE BETWEEN OLD & NEW
11	IE	4096	A value comparison was unequal
12	LB	4096	A value comparison was unequal
13	MH	4096	A value comparison was unequal
14	PE	4096	A value comparison was unequal
15	PR	4096	A value comparison was unequal
16	RELREC	4224	NEW data set has observation not in OLD, A value comparison was unequal
17	SC	4160	OLD data set has observation not in NEW, A value comparison was unequal
18	SE	4096	A value comparison was unequal
19	SU	4096	A value comparison was unequal
20	SV	4096	A value comparison was unequal
21	TA	0	NO DIFFERENCE BETWEEN OLD & NEW
22	TE	0	NO DIFFERENCE BETWEEN OLD & NEW
23	TI	32	Variable has different label
24	TV	0	NO DIFFERENCE BETWEEN OLD & NEW
25	VS	4096	A value comparison was unequal

CONCLUSION

We have demonstrated in this paper yet another technique for validating data sets underlying clinical trial data.

The use of the SAS® bitwise AND function enables an efficient interpretation of the SYSINFO return codes generated by PROC COMPARE, and allows the generation of compact comparison reports, as part of clinical data validation. This approach also enables the automatic insertion of reports into Excel workbook by outputting SYSINFO reports into Excel worksheets where a single worksheet represents an entire folder of SAS® data sets (not shown in this paper).

The SYSINFO approach further permits the programming of decision making when certain differences are detected among data sets. For instance, through programming logic, one can choose to ignore "cosmetic" differences between data sets. SYSINFO codes less than 64 usually relate to formatting and not considered critical.

Similarly, a comparison that generates SYSINFO=0 for data sets might not even need to generate reports, as the return code would indicate "no differences detected". A code of 32768 ("Fatal error") could be made to trigger the suspension of the running of the program, saving valuable time.

It is therefore apparent that SYSINFO coding and interpretation could be a powerful tool in the potentially arduous task of validating clinical data. The use of the bAND function makes the process even more efficient.

REFERENCES

Abraham, A. (2010) “A Few Quick and Efficient Ways to Compare Data”,
Proceedings of the North-East SAS® Users Group (NESUG) 2010.
http://www.nesug.org/Proceedings/nesug10/po/po27.pdf

Fennell, L. (2006) “An Easy, Concise Way to Summarize Multiple PROC COMPAREs Using the SYSINFO Macro Variable”,
Proceedings of the South-East SAS® Users Group (SESUG) 2006.
http://analytics.ncsu.edu/sesug/2006/SC21_06.PDF

Fogleman, S. (2006) “The Use of SAS® Literals and Why They are Still Relevant!”
Proceedings of the SAS® Global Forum (SGF) 2006.
http://www2.sas.com/proceedings/sugi31/026-31.pdf

Pohl, G. and Ye, W. (2007) “Efficient Storage and Processing of Sequential Indicators Using SAS® Bitwise Functions”,
Proceedings of the Pharmaceutical Industry SAS® Users Group (PharmaSUG) 2007.
http://www.lexjansen.com/pharmasug/2007/pr/pr01.pdf

ACKNOWLEDGMENTS

The authors would like to thank their management and colleagues who provided valuable input and comments on this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Joseph Hinson*
Merck & Co., Inc.
RY34-A320
P.O. Box 2000
Rahway, NJ 07065
Phone: 732-594-7789

E-mail: joseph.hinson@merck.com

Margaret M. Coughlin
Manager, D&E and Scientific Programming,
BARDS
Merck & Co., Inc.
RY34-A320
Rahway, NJ 07065
Phone: 732-594-3781
E-mail: margaret_coughlin@merck.com

* Joseph Hinson is working for Merck under a contract from Agile-1, 1999 West 190th St., Torrance, CA, 90504.