”I Want it _ALL_ and I want it now!”

Gary Hearfield
Head of International Statistical Programming, Cambridge, UK

Coders’ Corner – CC05

6th May 2009

Contents

1) Introduction: What is _ALL_?
2) Its role in handling Duplicate records
3) Its use within procedures and the data step
4) Closing Thoughts

The aim of this paper is to **highlight the role that _ALL_ can play** in simplifying the code required for everyday programming tasks with a focus on specific situations where it can **become a useful part of the programmers armory**

Introduction

- What is _ALL_?
  1) A “Keyword” (PUT _ALL_)
  2) A variable name list (BY _ALL_)
  3) A dataset name list (DATA=<*lib*>._ALL_)
  4) “Just do whatever on EVERYTHING” (ods _ALL_ close)
- Not very well documented (IMHO)
- Why don’t SAS have differently named statements for different functionality?

Duplicates or "Can you repeat that?" PhUSE 2006 (SS01) by Lawrence Heaton-Wright

USING DATA STEP BY-PROCESSING

To be absolutely certain which observation we are selecting as our unique record we can always use BY-processing within the DATA step which allows the usage of the FIRST and LAST processing available within the DATA step.

```
PROC SORT DATA=duplicate_example;
    BY subjid visit value;
RUN;

DATA firstdot_example;
    SET duplicate_example;
    BY subjid visit value;
    IF FIRST.visit;
RUN;
```

Obs	subjid	visit	value
1	1	1	87
2	1	2	79
3	1	3	85
4	1	4	91
5	1	5	87

Example Data

VIEWTABLE: Work.Testdata

	subjid	value	visit	seqno
1	1	87	1	1
2	1	79	2	1
3	1	86	3	1
4	1	85	3	1
5	1	86	3	2
6	1	91	4	1
7	1	87	5	1

DUPLICATE VALUES (within subjid and visit)

Or are they?

(what is this information?)

Eliminating duplicates using PROC SORT and "Key variables"

```
* Duplicates within KEY identifiers "Subjid" and "visit"? - Losing information ;
PROC SORT DATA=testdata OUT=example1 NODUPKEY ;
  BY subjid visit ;
RUN ;
```

```
NOTE: There were 7 observations read from the data set WORK.TESTDATA.
NOTE: 2 observations with duplicate key values were deleted.
NOTE: The data set WORK.EXAMPLE1 has 5 observations and 4 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time               0.20 seconds
      cpu time                0.00 seconds
```

VIEWTABLE: Work.Example1

	subjid	value	visit	seqno
1	1	87	1	1
2	1	79	2	1
3	1	86	3	1
4	1	91	4	1
5	1	87	5	1

```
* Any "TRUE" Duplicates across all variables? - No information lost ;
PROC SORT DATA=testdata OUT=example2 NODUPKEY ;
  BY subjid visit value ;
RUN ;
```

```
NOTE: There were 7 observations read from the data set WORK.TESTDATA.
NOTE: 1 observations with duplicate key values were deleted.
NOTE: The data set WORK.EXAMPLE2 has 6 observations and 4 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time               0.03 seconds
      cpu time                0.00 seconds
```

VIEWTABLE: Work.Example2

	subjid	value	visit	seqno
1	1	87	1	1
2	1	79	2	1
3	1	85	3	1
4	1	86	3	1
5	1	91	4	1
6	1	87	5	1

Eliminating duplicates using PROC SORT and _ALL_

```
* IS _ALL_ in the BY statement (= sort by all variables in the dataset) a possible answer?;
* No information lost - only records with identical values in ALL fields are removed ;
PROC SORT DATA=testdata OUT=example3 NODUPKEY ;
  BY _all_ ;
RUN ;
```

```
NOTE: There were 7 observations read from the data set WORK.TESTDATA.
NOTE: 0 observations with duplicate key values were deleted.
NOTE: The data set WORK.EXAMPLE3 has 7 observations and 4 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time               0.04 seconds
      cpu time                0.00 seconds
```

VIEWTABLE: Work.Example3

	subjid	value	visit	seqno
1	1	79	2	1
2	1	85	3	1
3	1	86	3	1
4	1	86	3	2
5	1	87	1	1
6	1	87	5	1
7	1	91	4	1

Notice that _ALL_ in the BY statement has had the effect of sorting BY subjid, value, visit and seqno in that order…Why is this?

```
* Use _ALL_ in the BY statement (= sort by all variables in the dataset) but DROP
  unrequired vars ;
* No information lost - only records with identical values in ALL fields KEPT are removed ;
PROC SORT DATA=testdata (DROP=seqno) OUT=example4 NODUPKEY ;
  BY _all_ ;
RUN ;
```

```
NOTE: There were 7 observations read from the data set WORK.TESTDATA.
NOTE: 1 observations with duplicate key values were deleted.
NOTE: The data set WORK.EXAMPLE4 has 6 observations and 3 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time               0.04 seconds
      cpu time                0.01 seconds
```

VIEWTABLE: Work.Example4

	subjid	value	visit
1	1	79	2
2	1	85	3
3	1	86	3
4	1	87	1
5	1	87	5
6	1	91	4

ALL in a PROC SORT BY statement (like PROC PRINT) handles variables in "variable number" order

The CONTENTS Procedure

Data Set Name	WORK.EXAMPLE4	Observations	6
Member Type	DATA	Variables	3
Engine	V9	Indexes	0
Created		Observation Length	24
Last Modified		Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	YES
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information

Data Set Page Size	4096
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	168
Obs in First Data Page	6
Number of Data Set Repairs	0
File Name	C:\DOCUME~1\ghearfie.EU\LOCALS~1\Temp\SAS Temporary Files_TD944\example4.sas7bdat
Release Created	9.0101M3
Host Created	WIN_PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	subjid	Num	8
2	value	Num	8
3	visit	Num	8

Sort Information

Sortedby	subjid value visit
Validated	YES
Character Set	ANSI
Sort Option	NODUPKEY

They SQL Here, They SQL There
PhUSE 2007 (TU05) Ian Amaranayake

```
      * Use PROC SQL to Re-order the variables within the dataset to allow control of _ALL_ ;
      PROC SQL;
        create table testdata2 as
        select subjid, visit, value
        from testdata
        order by subjid, visit, value ;
NOTE: Table WORK.TESTDATA2 created, with 7 rows and 3 columns.

      QUIT ;
NOTE: PROCEDURE SQL used (Total process time):
      real time               0.37 seconds
      cpu time                0.01 seconds


      * Check to confirm re-ordering ;
      PROC CONTENTS DATA=testdata2;
      RUN ;

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time               0.01 seconds
      cpu time                0.01 seconds
```

PROC SQL re-orders the variables within the dataset

The CONTENTS Procedure

Data Set Name	WORK.TESTDATA2	Observations	7
Member Type	DATA	Variables	3
Engine	V9	Indexes	0
Created		Observation Length	24
Last Modified		Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	YES
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information

Data Set Page Size	4096
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	168
Obs in First Data Page	7
Number of Data Set Repairs	0
File Name	C:\DOCUME~1\ghearfie.EU\LOCALS~1\Temp\SAS Temporary Files_TD944\testdata2.sas7bdat
Release Created	9.0101M3
Host Created	WIN_PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	subjid	Num	8
3	value	Num	8
2	visit	Num	8

Sort Information

Sortedby	subjid visit value
Validated	YES
Character Set	ANSI

Variables with data set re-numbered

```
      * Having ensure that all the variables are numbered as required, ;
      *   we can use have confidence in _ALL_ ;
      PROC SORT DATA=testdata2 OUT=example6 NODUPKEY ;
        BY _all_ ;
      RUN ;

NOTE: There were 7 observations read from the data set WORK.TESTDATA2.
NOTE: 1 observations with duplicate key values were deleted.
NOTE: The data set WORK.EXAMPLE6 has 6 observations and 3 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time               0.14 seconds
      cpu time                0.00 seconds


      DATA example7 ;
        SET example6 ;
        BY subjid visit ;
      RUN ;

NOTE: There were 6 observations read from the data set WORK.EXAMPLE6.
NOTE: The data set WORK.EXAMPLE7 has 6 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time               0.04 seconds
      cpu time                0.00 seconds


      DATA example8 ;
        SET example6 ;
        BY _all_ ;
      RUN ;

NOTE: There were 6 observations read from the data set WORK.EXAMPLE6.
NOTE: The data set WORK.EXAMPLE8 has 6 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time               0.00 seconds
      cpu time                0.00 seconds
```

	subjid	visit	value
1	1	1	87
2	1	2	79
3	1	3	85
4	1	3	86
5	1	4	91
6	1	5	87

NO ERRORS!

(so clearly sorted by subjid and visit)

ALL and PUT statements ("Keyword")

```
DATA example8 ;
  SET example6 ;
  BY _all_ ;

  PUT _ALL_ ;

  %PUT _ALL_ ;
GLOBAL SQLOOPS 21
GLOBAL SQLXOBS 0
GLOBAL SQLRC 0
AUTOMATIC AFDSID 0
AUTOMATIC AFDSNAME
AUTOMATIC AFLIB
AUTOMATIC AFSTR1
AUTOMATIC AFSTR2
AUTOMATIC FSPBDV
AUTOMATIC SYSBUFFR
AUTOMATIC SYSCC 0
AUTOMATIC SYSCHARWIDTH 1
AUTOMATIC SYSCMD

AUTOMATIC SYSUSERID ghearfie
AUTOMATIC SYSVER 9.1
AUTOMATIC SYSVLONG 9.01.01M3P020206
AUTOMATIC SYSVLONG4 9.01.01M3P02022006
RUN ;

subjid=1 visit=1 value=87 FIRST.subjid=1 LAST.subjid=0 FIRST.visit=1 LAST.visit=1 FIRST.value=1
LAST.value=1 _ERROR_=0 _N_=1
subjid=1 visit=2 value=79 FIRST.subjid=0 LAST.subjid=0 FIRST.visit=1 LAST.visit=1 FIRST.value=1
LAST.value=1 _ERROR_=0 _N_=2
subjid=1 visit=3 value=85 FIRST.subjid=0 LAST.subjid=0 FIRST.visit=1 LAST.visit=0 FIRST.value=1
LAST.value=1 _ERROR_=0 _N_=3
subjid=1 visit=3 value=86 FIRST.subjid=0 LAST.subjid=0 FIRST.visit=0 LAST.visit=1 FIRST.value=1
LAST.value=1 _ERROR_=0 _N_=4
subjid=1 visit=4 value=91 FIRST.subjid=0 LAST.subjid=0 FIRST.visit=1 LAST.visit=1 FIRST.value=1
LAST.value=1 _ERROR_=0 _N_=5
subjid=1 visit=5 value=87 FIRST.subjid=0 LAST.subjid=1 FIRST.visit=1 LAST.visit=1 FIRST.value=1
LAST.value=1 _ERROR_=0 _N_=6
NOTE: There were 6 observations read from the data set WORK.EXAMPLE6.
NOTE: The data set WORK.EXAMPLE8 has 6 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           0.04 seconds
      cpu time            0.00 seconds
```

ALL compared with _numeric_ and _character_ (similarities: when used as "variable lists")

```
DATA example8 ;
  SET example6 ;
  BY _all_ ;
  PUT (_ALL_) (=) ;
RUN ;
```

Parentheses turn _ALL_ from "keyword" to a variable list

```
subjid=1 visit=1 value=87
subjid=1 visit=2 value=79
subjid=1 visit=3 value=85
subjid=1 visit=3 value=86
subjid=1 visit=4 value=91
subjid=1 visit=5 value=87
NOTE: There were 6 observations read from the data set WORK.EXAMPLE6.
NOTE: The data set WORK.EXAMPLE8 has 6 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Data set variables only (NOT automatic variables)

```
DATA example9 ;
  SET example6 ;
  BY _all_ ;
  PUT (_character_) (=) ;
  PUT (_numeric_) (=) ;
RUN ;
```

```
subjid=1 visit=1 value=87

subjid=1 visit=2 value=79

subjid=1 visit=3 value=85

subjid=1 visit=3 value=86

subjid=1 visit=4 value=91

subjid=1 visit=5 value=87
NOTE: There were 6 observations read from the data set WORK.EXAMPLE6.
NOTE: The data set WORK.EXAMPLE9 has 6 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

ALL compared with _numeric_ and _character_ (Differences: _ALL_ can be used as a "keyword")

```
DATA example10 ;
  SET example6 ;
  BY _all_ ;
  PUT _character_ ;
RUN ;
```

```
NOTE: Variable _character_ is uninitialized.
```

```
.
.
.
.
.
.
```

```
DATA example10 ;
  SET example6 ;
  BY _all_ ;
  PUT _numeric_ ;
ERROR: Cannot use _numeric_ as a variable name.
RUN ;
```

```
NOTE: The SAS System stopped processing this step because of errors.
```

ALL and arrays (variable list)

```
     DATA example11 ;
       SET example6 ;
       BY _all_ ;
       ARRAY all [*] _all_ ;
       ARRAY num [*] _numeric_ ;
       ARRAY char [*] _chracter_ ;
     RUN ;

NOTE: There were 6 observations read from the data set WORK.EXAMPLE6.
NOTE: The data set WORK.EXAMPLE11 has 6 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time           0.13 seconds
      cpu time            0.00 seconds



      DATA example12 ;
        SET example6 ;
        BY _all_ ;
        testtype = "Result" ;
        ARRAY all [*] _all_ ;
ERROR: All variables in array list must be the same type, i.e., all numeric or character.
      RUN ;

NOTE: The SAS System stopped processing this step because of errors.
```

ALL with *LIBREFS* and *FILEREFS* (Just send information to the .LOG)

```
FILENAME testdata "work.testdata" ;

FILENAME _all_ list ;
NOTE: Fileref= TESTDATA
      Physical Name= C:\Documents and Settings\ghearfie.EU\work.testdata

FILENAME _all_ clear ;
NOTE: Fileref TESTDATA has been deassigned.
```

```
LIBNAME _all_ list ;
NOTE: Libref=    SASHELP
      Scope=     Kernel
      Levels=    14
        -Level 1-
      Engine=    V9
      Physical Name= C:\Program Files\SAS\SAS 9.1\nls\en\SASCFG
      File Name= C:\Program Files\SAS\SAS 9.1\nls\en\SASCFG
        -Level 2-
      Engine=    V9
      Physical Name= C:\Program Files\SAS\SAS 9.1\core\sashelp
      File Name= C:\Program Files\SAS\SAS 9.1\core\sashelp
        -Level 3-
      Engine=    V9
      Physical Name= C:\Program Files\SAS\SAS 9.1\af\sashelp
      File Name= C:\Program Files\SAS\SAS 9.1\af\sashelp
        -Level 4-
      Engine=    V9
      Physical Name= C:\Program Files\SAS\SAS 9.1\assist\sashelp
      File Name= C:\Program Files\SAS\SAS 9.1\assist\sashelp
        -Level 5-
      Engine=    V9
      Physical Name= C:\Program Files\SAS\SAS 9.1\connect\sashelp
      File Name= C:\Program Files\SAS\SAS 9.1\connect\sashelp
        -Level 6-
      Engine=    V9
      Physical Name= C:\Program Files\SAS\SAS 9.1\eis\sashelp
      File Name= C:\Program Files\SAS\SAS 9.1\eis\sashelp
        -Level 7-
      Engine=    V9
      Physical Name= C:\Program Files\SAS\SAS 9.1\ets\sashelp
      File Name= C:\Program Files\SAS\SAS 9.1\ets\sashelp
        -Level 8-
      Engine=    V9
      Physical Name= C:\Program Files\SAS\SAS 9.1\graph\sashelp
      File Name= C:\Program Files\SAS\SAS 9.1\graph\sashelp
        -Level 9-
      Engine=    V9
      Physical Name= C:\Program Files\SAS\SAS 9.1\iml\sashelp
      File Name= C:\Program Files\SAS\SAS 9.1\iml\sashelp
```

ALL and PROC CONTENTS (Data Set name list: send info to Output window)

```
     PROC CONTENTS DATA=work._all_ ;
     RUN ;

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time           0.45 seconds
      cpu time            0.03 seconds


                              The CONTENTS Procedure

                                   Directory

      Libref          WORK
      Engine          V9
      Physical Name   C:\DOCUME~1\ghearfie.EU\LOCALS~1\Temp\SAS Temporary Files\_TD944
      File Name       C:\DOCUME~1\ghearfie.EU\LOCALS~1\Temp\SAS Temporary Files\_TD944

                                 Member   File
                    #  Name      Type     Size  Last Modified

                    1  EXAMPLE1  DATA     5120
                    2  EXAMPLE10 DATA     5120
                    3  EXAMPLE11 DATA     5120
                    4  EXAMPLE12 DATA     5120
                    5  EXAMPLE2  DATA     5120
                    6  EXAMPLE3  DATA     5120
                    7  EXAMPLE4  DATA     5120
                    8  EXAMPLE5  DATA     5120
                    9  EXAMPLE6  DATA     5120
                   10  EXAMPLE7  DATA     5120
                   11  EXAMPLE8  DATA     5120
                   12  EXAMPLE9  DATA     5120
                   13  TESTDATA  DATA     5120
                   14  TESTDATA2 DATA     5120
```

ALL and PROC DATASETS (Variable or Data Set name list)

```
PROC DATASETS library = work ;
                                        Directory

   Libref          WORK
   Engine          V9
   Physical Name   C:\DOCUME~1\ghearfie.EU\LOCALS~1\Temp\SAS Temporary Files\_TD944
   File Name       C:\DOCUME~1\ghearfie.EU\LOCALS~1\Temp\SAS Temporary Files\_TD944

                                Member   File
                      #  Name   Type     Size  Last Modified

                      1  EXAMPLE1   DATA     5120
                      2  EXAMPLE10  DATA     5120
                      3  EXAMPLE11  DATA     5120
                      4  EXAMPLE12  DATA     5120
                      5  EXAMPLE2   DATA     5120
                      6  EXAMPLE3   DATA     5120
                      7  EXAMPLE4   DATA     5120
                      8  EXAMPLE5   DATA     5120
                      9  EXAMPLE6   DATA     5120
                     10  EXAMPLE7   DATA     5120
                     11  EXAMPLE8   DATA     5120
                     12  EXAMPLE9   DATA     5120
                     13  TESTDATA   DATA     5120
                     14  TESTDATA2  DATA     5120
   contents data=_ALL_ nods ;
   modify testdata ;
     attrib _all_ label = '' '' ; * Remove labels ;
     format _all_ ;               * Remove formats ;
   quit ;
NOTE: MODIFY was successful for WORK.TESTDATA.DATA.
NOTE: PROCEDURE DATASETS used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds


RUN ;
```

Closing Thoughts

- _ALL_ is everywhere in SAS
- _ALL_ **is what it does** and has various different functionality in different situations
- _ALL_ has the potential to save time and reduce amount of code

Contact Details

Gary Hearfield

Head of International Statistical Programming

Amgen Limited

240 Cambridge Science Park

Milton Road

Cambridge

CB4 0WD

Tel: **+44 (0) 1223 436314**

Mob: **+44 (0) 7960 310809**

Email: **GHearfie@Amgen.com**

Questions?????

- Are duplicate records a problem in your organisation?
- Do you have in-house built tools to deal with this issue?
- Do you have a better method?
- Should SAS have built in functionality (PROC COMPARE)?