Paper 204-25

Proc Doc

A SAS® Software Prescription for Your Documentation Ills

Louise Hadden
Abt Associates Inc., Cambridge, MA

ABSTRACT

The most onerous task any SAS programming professional faces is to accurately document files and processes. By setting careful standards at the outset of a programming task and using the many tools the SAS® system provides to assist in the documentation process, document files and processes need not be "the dreaded D word".

WHY DOCUMENT?

Documenting files and processes is important for a number of reasons. First, most organizations have a system that ensures quality products. Such a system may include a third party attempting to replicate results, reviewing code, etc. To facilitate a review of a SAS programmer's product, it is useful to be able to provide accurate and clear documentation of both data files produced and utilized, and the SAS programs that create and analyze data. Secondly, many programmers reuse code and/or data files over months and years. Well-documented files (code and data) are easier to review for subsequent use, particularly if the version of SAS at a given site has changed over time (or if you have changed sites and taken your code with you!), and for Y2K compliance issues. Last but not least, it may be necessary to repeat certain operations in the event of a disk crash, media failure or other such calamity. Of course, we all have timed backups at least once a day so that shouldn't be a problem, but . . . it is much easier to reproduce files and operations with accurate documentation.

START AT THE BEGINNING AND TAKE YOUR MEDICINE!

The easiest way to produce good documentation is to start with the programs that create or use given files. It is useful to start with a boiler plate "doc box" which provides such information as the program name, the program author, the original program date, a short description of the intended use of the program, revision date(s), inputs and outputs including macro catalogs, format libraries and graphic templates used, platform, and location of the program file. I also include a note at the top indicating that the code has been checked for Y2K compliance. A comment line may be added to note anything unusual or usual that might be useful information.

```
**********************************;
*   Checked for Y2K Compliance
*   program:       CONTENT1.SAS
*   date written:  XX/XX/XXXX
*   date revised:  XX/XX/XXXX
*   author:        LSH
*   location:      xxxxxxxx/hadden
*   platform:      RS/6000
*   input(s):      XXXXXXXX Files
*   output(s):     Printout,
                   flat ASCII files
*   description:   Produce proc
                   contents on
                   various data
                   sets and
                   converts output
                   into flat files
**********************************;
```

Following the "doc box", titles and footnotes can be used to provide useful information. I usually reserve TITLE1 for the contract the program is being used in the performance of. TITLE2 includes the name of the data file(s) being created and used. The footnote notes the program name (with full path) and the macro variable &SYSDATE9. which indicates when the program was last run (particularly useful for times when the NODATE system option is used). Other similar macro variables can be used as well, indicating the platform or version of SAS being used.

```
title1 'Contract XXX';
title2 'File:  YYY';
footnote "Program: e6/hadden/content1.sas -
Last Run &sysdate9.";
```

Whether creating a new version of existing file(s) or creating a SAS data set from external data, the SAS system provides many ways to include information about data sets and variables. The use of the label data set option allows the SAS programmer to specify useful information about the data set(s) such as the program(s) used to create or modify

the data set(s). The religious use of label and format statements provides additional information to SAS programmers.

Last but not least, commenting code consistently is the single best way to document processes. This is particularly important in this era of FSEDIT, the DDE triplet, OBDC and interactive processing. As well as ensuring readability and reproducible results, commenting code adds an extra measure of thought toward checking logic in complex operations.

```
%macro readit(year,qtr,filenum);

***Data Step***;

data yr&year._q&qtr. (drop=I);
    length statements;
    infile zip&filenum lrecl=388;

***input the variables from the ASCII file***;

    input variables @;

***select records based on values of variables***;

    if prim=1 or second=1
        then delete;

    input more;

    more SAS code . . .

***label variables***;

    label x='XXX';
run;

***use proc datasets to remove preliminary quarterly
files and append quarterly files to yearly data set
as they are created***

proc datasets library=dd;
    append base=hcup19&yr
    data=yr&yr._q&qtr;
    delete yr&yr._q&qtr.;
run;

%mend readit;
```

It is also possible to add in such useful information as data set labels, variable labels, length and formats after the fact. If you find a data set does not have complete information (a quick proc contents will make it obvious), you can fill it in using attrib, label, length and format statements (provided you can obtain the necessary information!)

LET PROC CONTENTS EASE YOUR PAIN

The lowly PROC CONTENTS can do amazing things. It is the single best source of information about a data set. Most SAS programmers are familiar with the procedure and use it extensively in its usual form, but fewer output the resulting data set and make use of it.

The output SAS data set produced by PROC CONTENTS when you specify the OUT= option contains information on the variables in the SAS data sets specified in the DATA= option. The variables in the output data set are described below.

CHARSET the value in the Character Set field.
COLLATE the value in the Collating Sequence field.
COMPRESS the value in the Compressed field.
CRDATE date the data set was created.
DELOBS number of observations marked for deletion in the data set.
ENGINE name of the method used to read from and write to the data set.
FORMAT variable format (blank if none given).
FORMATD number of decimals for format (0 if none given).
FORMATL format width (0 if none given).
IDXCOUNT number of indexes for the data set.
IDXUSAGE use of the variable in indexes.
INFORMAT variable informat (blank if none given).
INFORMD number of decimals for informat (0 if none given).
INFORML informat width (0 if none given).
JUST justification (0=left, 1=right).
LABEL variable label (blank if none given).
LENGTH variable length.
LIBNAME libref used for the data library.
MEMLABEL label for this data set (blank if no label).
MEMNAME member in which the variable is located.
MEMTYPE library member type (DATA or VIEW).
MODATE date the data set was last modified.
NAME variable name.
NOBS number of observations in the data set.
NODUPKEY the value is YES if you use this option in the PROC SORT statement, NO if you do not or if the data set is not sorted.
NODUPREC the value is YES if you use this option in the PROC SORT statement, NO if you do not or if the data set is not sorted.
NPOS relative position of the variable in the data set input buffer.
PROTECT the first letter of the level of protection shown in the Protection field.
REUSE the value of the Reuse Space field.
SORTED the value depends on the sorting characteristics of the input data set.
SORTEDBY the value depends on that variable's role in the sort.
TYPE type of the variable (1=numeric, 2=character).
TYPEMEM special data set type (blank if no TYPE= value is specified).
VARNUM variable number in the data set.

(SAS Online Documentation)

The challenge then becomes to utilize the extensive information provided on specific data sets in the most efficient and complete way possible. The most information will be gained from the data set that is labeled and documented prior to running the procedure.

The information provided in the output data set is on two levels; the data set level, and the variable level. An example of a listing from printing out the output data set from proc contents, and the results of running a proc contents on the output data set follow below.

```
Print Out Output File from Proc Contents

OBS LIBNAME  MEMNAME         MEMLABEL          TYPEMEM  NAME

 1    DD     SSIDATA  SSI Data, January 1999            AGE
 2    DD     SSIDATA  SSI Data, January 1999            FNAME
 3    DD     SSIDATA  SSI Data, January 1999            FULLADDR
 4    DD     SSIDATA  SSI Data, January 1999            LNAME
 5    DD     SSIDATA  SSI Data, January 1999            MANUMBER

OBS TYPE  LENGTH  VARNUM  LABEL                FORMAT  FORMATL

 1    1      8       4    SSI: Age                        0
 2    2     10       6    SSI: First Name                 0
 3    2     50       1    SSI: Full Address               0
 4    2     19       5    SSI: Last Name                  0
 5    2      8       3    SSI: Medicaid ID                0

OBS FORMATD INFORMAT INFORML INFORMD JUST NPOS NOBS ENGINE

 1     0                0       0      1    67 3054  V612
 2     0                0       0      0    94 3054  V612
 3     0                0       0      0     0 3054  V612
 4     0                0       0      0    75 3054  V612
```

```
 5     0               0       0      0    59 3054  V612

OBS           CRDATE           MODATE DELOBS IDXUSAGE MEMTYPE

 1  04JUL99:13:16:49 04JUL99:13:16:50    0     NONE    DATA
 2  04JUL99:13:16:49 04JUL99:13:16:50    0     NONE    DATA
 3  04JUL99:13:16:49 04JUL99:13:16:50    0     NONE    DATA
 4  04JUL99:13:16:49 04JUL99:13:16:50    0     NONE    DATA
 5  04JUL99:13:16:49 04JUL99:13:16:50    0     NONE    DATA


OBS IDXCOUNT PROTECT COMPRESS REUSE SORTED SORTEDBY CHARSET

 1      0      ---      NO     NO      .       .
 2      0      ---      NO     NO      .       .
 3      0      ---      NO     NO      .       .
 4      0      ---      NO     NO      .       .
 5      0      ---      NO     NO      .       .

OBS COLLATE    NODUPKEY    NODUPREC    ENCRYPT

 1                NO          NO          NO
 2                NO          NO          NO
 3                NO          NO          NO
 4                NO          NO          NO
 5                NO          NO          NO

OBS LIBNAME  MEMNAME         MEMLABEL         TYPEMEM  NAME

 6    DD     SSIDATA  SSI Data, January 1999           MI
 7    DD     SSIDATA  SSI Data, January 1999           SSN

OBS TYPE  LENGTH  VARNUM  LABEL                FORMAT  FORMATL

 6    2      1       7    SSI: Middle Initial             0
 7    2      9       2    SSI: SSN                        0

OBS FORMATD INFORMAT INFORML INFORMD JUST NPOS NOBS ENGINE

 6     0                0       0      0   104 3054  V612
 7     0                0       0      0    50 3054  V612

OBS           CRDATE           MODATE DELOBS IDXUSAGE MEMTYPE

 6  04JUL99:13:16:49 04JUL99:13:16:50    0     NONE    DATA
 7  04JUL99:13:16:49 04JUL99:13:16:50    0     NONE    DATA

OBS IDXCOUNT PROTECT COMPRESS REUSE SORTED SORTEDBY CHARSET

 6      0      ---      NO     NO      .       .
 7      0      ---      NO     NO      .       .

OBS COLLATE    NODUPKEY    NODUPREC    ENCRYPT

 6                NO          NO          NO
 7                NO          NO          NO


Print Out Contents of Output File from Proc Contents

CONTENTS PROCEDURE

Data Set Name: WORK.FLAT                Observations:         7
Member Type:   DATA                     Variables:            35
Engine:        V612                     Indexes:              0
Created:       13:23 Sun, Jul 4, 1999   Observation Length:   325
Last Modified: 13:23 Sun, Jul 4, 1999   Deleted Observations: 0
Protection:                             Compressed:           NO
Data Set Type:                          Sorted:               YES
Label:

-----Engine/Host Dependent Information-----

Data Set Page Size:       10240
Number of Data Set Pages: 1
File Format:              607
First Data Page:          1
Max Obs per Page:         31
Obs in First Data Page:   7

      -----Alphabetic List of Variables and Attributes-----

 # Variable Type Len Pos Format      Label
-----------------------------------------------------------------
31 CHARSET  Char    8 295             Host Character Set
32 COLLATE  Char    8 303             Collating Sequence
27 COMPRESS Char    8 268             Compression Routine
20 CRDATE   Num     8 216 DATETIME16. Create Date
22 DELOBS   Num     8 232             Deleted Observations in Data
                                      Set
35 ENCRYPT  Char    8 317             Encryption Routine
19 ENGINE   Char    8 208             Engine Name
10 FORMAT   Char    8 136             Variable Format
12 FORMATD  Num     8 152             Number of Format Decimals
11 FORMATL  Num     8 144             Format Length
25 IDXCOUNT Num     8 257             Number of Indexes for Data
                                      Set
23 IDXUSAGE Char    9 240             Use of Variable in Indexes
13 INFORMAT Char    8 160             Variable Informat
15 INFORMD  Num     8 176             Number of Informat Decimals
14 INFORML  Num     8 168             Informat Length
16 JUST     Num     8 184             Justification
 9 LABEL    Char   40  96             Variable Label
 7 LENGTH   Num     8  80             Variable Length
 1 LIBNAME  Char    8   0             Library Name
 3 MEMLABEL Char   40  16             Data Set Label
 2 MEMNAME  Char    8   8             Library Member Name
24 MEMTYPE  Char    8 249             Library Member Type
21 MODATE   Num     8 224 DATETIME16. Last Modified Date
 5 NAME     Char    8  64             Variable Name

18 NOBS     Num     8 200             Observations in Data Set
33 NODUPKEY Char    3 311             Sort Option: No Duplicate
                                      Keys
34 NODUPREC Char    3 314             Sort Option: No Duplicate
                                      Records
17 NPOS     Num     8 192             Position in Buffer
26 PROTECT  Char    3 265             Password Protection (Read
                                      Write Alter)
28 REUSE    Char    3 276             Reuse Space
29 SORTED   Num     8 279             Sorted and/or Validated
30 SORTEDBY Num     8 287             Position of Variable in
                                      Sortedby Clause
 6 TYPE     Num     8  72             Variable Type
 4 TYPEMEM  Char    8  56             Special Data Set Type (From
                                      TYPE=)
 8 VARNUM   Num     8  88             Variable Number


  -----Sort Information-----

Sortedby:      LIBNAME MEMNAME
Validated:     YES
Character Set: ANSI
```

The variables available in the output data set from PROC CONTENTS vary from platform to platform. If you cannot locate documentation specific to your platform, running a PROC CONTENTS with the output option, printing a sample of the resulting file, and running a PROC CONTENTS on the output file can be very useful. The example provided was run on the WINDOWS platform using SAS version 6.12.

In the macro provided below, two separate flat files are produced to incorporate into the documentation package, one for the data set as a whole, and one on the variable level.

```
%macro cont(infi,outfi,headfi,tit);

/* produce proc contents of data set with an
output file and a print file */

proc contents data=dd.&infi. position
out=temp;
title2 "Contents of &tit.";
run;

proc means data=temp noprint;
    var varnum;
    output out=tot max=nvars;
run;

/* create number of variables */

data temp;
    set temp;
    if _n_=1 then set tot (keep=nvars);
run;

/* sort output file by variable number */

proc sort data=temp;
    by varnum;
run;

/* set observations to 1 to avoid
duplication of data set level information */

options obs=1;

/* output flat file of data set level
information */

data _null_;
    set temp (keep=libname memname
    memlabel memtype compress
    reuse idxusage nvars
    crdate modate engine idxcount
    nobs nodupkey noduprec sorted);
    file &headfi. lrecl=80
    n=7;
```

```
    if sorted=1 then sort='YES';
    else sort='NO';
    format crdate modate datetime16.;

    put
        #1
        @1 'Data Set Information'
        #2
        @1 '&&'
        @5 "Member Name: " memname
        #3
        @1 '&&'
        @5 "Member Label: " memlabel
        #4
        @1 '&&'
        @5 "Member Type: " memtype
        @35 '@@'
        @40 "Compressed? " compress
        @60 '%%'
        @65 "Engine: " engine
        #5
        @1 '&&'
        @5 "Created: " crdate
        @35 '@@'
        @40 "Last Modified: "  modate
        #6
        @1 '&&'
        @5 "Number of Variables: " nvars
        @35 '@@'
        @40 "Number of Observations: " nobs
        #7
        @1 '&&'
        @5 "Sorted? " sort
        @35 '@@'
        @40 "NODUPKEY? " nodupkey
        @60 '%%'
        @65 "NODUPREC? " noduprec
        ;

run;

/* set observations to maximum to include all
variable level information */

options obs=max;

data _null_;
    set temp (keep=varnum name
    label type length npos sortedby
    just informat informl format
    formatl);
    file &outfi. lrecl=80 n=4;

    if type=2 then vartype='Char';
    else if type=1 then
    vartype='Num';
    if sortedby ne '.' and sortedby ne '0' then
sortvar='YES';
    else sortvar='NO';
    if just ne . and just ne 0 then justed='YES';
    else justed='NO';

/* output flat file of variable level information */

put     #1
        @1 '&&'
        @5 name
        @13 '@@'
        @15 label
        @55 '%%'
        @57 vartype
        @65 '##'
        @70 length
        #2
        @1 '&&'
        @5 "Var #: " varnum
        @13 '@@'
        @15 "Position: " npos
        @35 '%%'
        @40 "Sort Variable? " sortvar
        #3
        @1 '&&'
        @5 "Format (if any): " format
        @55 '%%'
        @57 "Informat (if any): " informat
        #4
    ;
run;

%mend;
```

Examples of the flat files produced by the above macro follow. Note that due to the column width restrictions for the paper that the output wraps. Obviously this would not normally be a problem. Characters such as &, % and @ are used to set tabs in word processing packages with proportional fonts.

Header Data:

```
Data Set Information
&&  Member Name: SSIDATA
&&  Member Label: SSI Data, January 1999
&&  Member Type: DATA              @@
Compressed? NO      %%   Engine: V612
&&  Created: 04JUL99:13:16:49     @@   Last
Modified: 04JUL99:13:16:50
&&  Number of Variables: 7        @@
Number of Observations: 3054
&&  Sorted? NO                    @@
NODUPKEY? NO        %%   NODUPREC? NO
```

Post Word-Processing Macro:

Data Set Information

Member Name: SSIDATA
Member Label: SSI Data, January 1999
Member Type: DATA Compressed? NO Engine: V612
Created: 04JUL99:13:16:49 Last Modified: 04JUL99:13:16:50
Number of Variables: 7 Number of Observations: 3054
Sorted? NO NODUPKEY? NO NODUPREC? NO

Variable Level Data:

```
&&  FULLADDR@@SSI: Full Address
%%Char    ##   50
&&  Var #: 1@@Position: 0         %%   Sort
Variable? NO
&&  Format (if any):
%%Informat (if any):

&&  SSN     @@SSI: SSN
%%Char    ##   9
&&  Var #: 2@@Position: 50        %%   Sort
Variable? NO
&&  Format (if any):
%%Informat (if any):

&&  MANUMBER@@SSI: Medicaid ID
%%Char    ##   8
&&  Var #: 3@@Position: 59        %%   Sort
Variable? NO
```

```
&&  Format (if any):
%%Informat (if any):

&&  AGE      @@SSI: Age
%%Num      ##   8
&&  Var #: 4@@Position: 67        %%   Sort
Variable? NO
&&  Format (if any):
%%Informat (if any):

&&  LNAME    @@SSI: Last Name
%%Char     ##   19
&&  Var #: 5@@Position: 75        %%   Sort
Variable? NO
&&  Format (if any):
%%Informat (if any):

&&  FNAME    @@SSI: First Name
%%Char     ##   10
&&  Var #: 6@@Position: 94        %%   Sort
Variable? NO
&&  Format (if any):
%%Informat (if any):

&&  MI       @@SSI: Middle Initial
%%Char     ##   1
&&  Var #: 7@@Position: 104       %%   Sort
Variable? NO
&&  Format (if any):
%%Informat (if any):
```

Post Word-Processing Macro:

FULLADDR	SSI: Full Address	Char50
Var #: 1	Position: 0	Sort Variable? NO
Format (if any):		Informat (if any):
SSN	SSI: SSN	Char9
Var #: 2	Position: 50	Sort Variable? NO
Format (if any):		Informat (if any):
MANUMBER	SSI: Medicaid ID	Char8
Var #: 3	Position: 59	Sort Variable? NO
Format (if any):		Informat (if any):
AGE	SSI: Age	Num8
Var #: 4	Position: 67	Sort Variable? NO
Format (if any):		Informat (if any):
LNAME	SSI: Last Name	Char19
Var #: 5	Position: 75	Sort Variable? NO
Format (if any):		Informat (if any):
FNAME	SSI: First Name	Char10
Var #: 6	Position: 94	Sort Variable? NO
Format (if any):		Informat (if any):
MI	SSI: Middle Initial	Char1
Var #: 7	Position: 104	Sort Variable? NO
Format (if any):		Informat (if any):

THE WONDERS OF %SYSFUNC

%SYSFUNC is a very useful macro function that became available in SAS release 6.12. Programmers now have access to most data set functions and many SCL functions via the macro processor. %SYSFUNC can provide the same information to the programmer as PROC CONTENTS, and more. While some of these additional uses of %SYSFUNC, such as checking for the existence of and manipulating external files, can be relevant to the SAS programmer and documentation efforts, this paper will address information that can be obtained on SAS data sets using %SYSFUNC. Below follows a simple macro that produces flat files similar to the PROC CONTENTS method described above.

```
%macro docds (infi,outfi);

/* open the data set */

%let ds_id=%SYSFUNC(open(&infi, i ));

data _null_;
    file &outfi;

/* header information */

%let totobs=%SYSFUNC(attrn(&ds_id, NOBS));

%let totvars=%SYSFUNC(attrn(&ds_id, NVARS));

%let credate=%SYSFUNC(attrn(&ds_id,
CRDTE),datetime18.);

%let moddate=%SYSFUNC(attrn(&ds_id,
MODTE),datetime18.);

%let engine=%SYSFUNC(attrc(&ds_id, ENGINE));


put / "Data Set Information";
put / "File Name=&infi";
put / "Total Observations in &infi=&totobs";
put / "Number of Variables in
&infi=&totvars";
put / "Created on:  &credate";
put / "Last Modified on:  &moddate";
put / "Engine:  &engine";
put / ;
put / "Variable Level Information";

/* variable level information */

%do i=1 %to &totvars;
    %let varnm&i=
        %SYSFUNC(varname(&ds_id,&i));
    put / "Variable &i Name=&&varnm&i";
    %let varlb&i=
        %SYSFUNC(varlabel(&ds_id,&i));
    put / "Variable &i Label=&&varlb&i";
%end;

/* close data set */

rc=%SYSFUNC(close(&ds_id));

return;

run;

%mend docds;
```

Note that the functions available to %SYSFUNC vary by platform. For example, our RS/6000's version 6.12 SAS allows the use of SORTEDBY while our Windows version 6.12 SAS does not. Consult the SAS Companion and online documentation for details on your particular operating system.

Example of output from the macro process above:

```
Data Set Information

File Name=DD.SSIDATA

Total Observations in DD.SSIDATA=3054

Number of Variables in DD.SSIDATA=7

Created on:  04JUL99:13:16:49

Last Modified on:  04JUL99:13:16:50

Engine:  V612


Variable Level Information

Variable 1 Name=FULLADDR

Variable 1 Label=SSI: Full Address

Variable 2 Name=SSN

Variable 2 Label=SSI: SSN

Variable 3 Name=MANUMBER

Variable 3 Label=SSI: Medicaid ID

Variable 4 Name=AGE

Variable 4 Label=SSI: Age

Variable 5 Name=LNAME

Variable 5 Label=SSI: Last Name

Variable 6 Name=FNAME

Variable 6 Label=SSI: First Name

Variable 7 Name=MI

Variable 7 Label=SSI: Middle Initial
```

THE SPOONFUL OF SUGAR

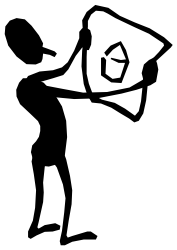
The flat files produced by the two simple macros described above can easily be incorporated into word processing documents and spreadsheets. The files may have special character strings inserted in records to enable the use of word processing packages to convert the characters into tabs and returns . If you are producing flat files for use on systems that do not have proportional fonts (or you don't mind living with Courier) these special characters can be omitted from the put statements. I also use a word processor to retain only the 'doc box' and comment lines in a program to incorporate into my documentation package for a contract. Another useful 'trick' is to name any user defined formats with its matching variable name, and output files with the PROC FORMAT CNTLOUT option and put statements to include in your documentation, if so desired.

CONCLUSION

The SAS system provides numerous opportunities for creating self-documenting data sets. With due diligence at the outset of a programming project, SAS tools such as PROC CONTENTS and %SYSFUNC can write the prescription for your documentation ills.

REFERENCES

SAS Language, Version 6, First Edition
SAS Procedures, Version 6, Third Edition
SAS Companion for the UNIX environment: Language, Version 6
SAS Companion for the Microsoft Windows Environment
SAS Macro Facility Tips and Techniques
SAS Online Documentation (PC SAS V612, AIX UNIX SAS V612)



ACKNOWLEDGMENTS

The author wishes to thank Chris Yindra for his inspirational paper and presentation on %SYSFUNC.

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

CONTACT INFORMATION

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

Louise Hadden
Abt Associates Inc.
55 Wheeler St.
Cambridge, MA 02138
Work Phone: 617-349-2385
Fax: 617-349-2675

Email: louise_hadden@abtassoc.com

KEYWORDS

SAS, PROC CONTENTS, Macro Language, Documentation, %SYSFUNC