

PROC FORMAT

1. beginning basics
2. power practices
3. where warning

by Doug McAllaster

for DC SAS User Group

on 11 March 2008

`douglas.mcallaster@tma.osd.mil`

`douglas.mcallaster@us.army.mil`

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.

PROC FORMAT

→ beginning basics

2. power practices

3. where warning

SAS has two data types: 1) character & 2) numeric

proc format has two “main” statements: 1) value & 2) invalue

1. A *value* statement creates a *format* which
converts vars when used in a *put* function

(a) numb \rightarrow char (type=N) *

(b) char \rightarrow char (type=C)

2. An *INvalue* statement creates an *INformat* which
converts vars when used in an *INput* function

(a) char \rightarrow numb (type=I) *

(b) char \rightarrow char (type=J)

proc format makes a catalog w/ entry types & an implied prefix:

1. value statement – formats

(a) numb → char (type=N)

et=format, implied fmtname prefix=none

(b) char → char (type=C)

et=formatC, implied fmtname prefix=\$

2. invalue statement – informats

(a) char → numb (type=I)

et=infmt, implied fmtname prefix=@

(b) char → char (type=J)

et=infmtC, implied fmtname prefix=@\$

However, since proc format can read & write tables containing the “mapping” info,

I prefer to manage input tables rather than catalog entries.

Others prefer using the `fmtsearch=(paths)` system option.

proc format can:

read cntLIN table – table → catalog entry

write cntLOUT table – catalog entry → table

the key columns (vars) in cntLIN,cntLOUT tables are:

1. fmtname - (without special prefixes, if type is used)
2. type = N, C, I, J
3. start - conversion from (start of range) – required
4. end - conversion from (end of range, or if omitted, same as start)
5. label - conversion to – required
6. hlo = blank, H, L, O: normal, High, Low, & Other)

I recommend using all six vars (or five excluding “end”) vars.
Well, enough discussion, let’s get to some concrete examples.

some data

```
> data d1;  
> length mf          $1 cod1          3 name $5;  
> length mforig $1 cod1orig 3 name $5;  
> input  mf          cod1          name @@;  
> mforig=mf;          cod1orig=cod1;  
> datalines;
```

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.

NOTE: The data set WORK.D1 has 6 observations and 5 variables.

```
> ; run;  
> proc print data=d1; run;
```

NOTE: There were 6 observations read from the data set WORK.D1.

some data

Obs	mf	cod1	name	mforig	cod1orig
1	M	10	tom	M	10
2	M	10	dick	M	10
3	M	10	harry	M	10
4	F	20	jane	F	20
5	F	20	sue	F	20
6	F	20	barb	F	20

cntlout shows 4 types of fmt/infmt

```
> proc format;
>   value   /* numb to char (prefix=null ) */
>     abc 10=men 20=women other=else      ;
NOTE: Format ABC has been output.
>   value   /* char to char (prefix=dollar) */
>     $abc M=male F=female other=unkn     ;
NOTE: Format $ABC has been output.
>   invalue /* char to numb (prefix=@ ) */
>     abc M=1 F=2 other=_error_ ;
NOTE: Informat ABC has been output.
>   invalue /* char to char (prefix=@$ ) */
>     $abc M=man F=woman other=_error_ ;
NOTE: Informat $ABC has been output.
> run;
> proc format
>   cntlout=f1 (keep=fmtname type start label hlo);
> run;
NOTE: The data set WORK.F1 has 12 observations and 5 variables.
> proc print data=f1 u; var start label hlo;
> by fmtname type notsorted; id fmtname type; run;
NOTE: There were 12 observations read from the data set WORK.F1.
```

type=N numb → char

sav1

```
-----  
|          FORMAT NAME: ABC          LENGTH:      5  NUMBER OF VALUES:      3  
|  MIN LENGTH:      1  MAX LENGTH:  40  DEFAULT LENGTH  5  FUZZ: STD  
|-----  
| START          | END          | LABEL (VER. V7|V8  10MAR2008:08:20:27)  
|-----+-----+-----  
|          10 |          10 | men  
|          20 |          20 | women  
| **OTHER**   | **OTHER**   | else  
-----
```

type=C char → char

```
-----  
|          FORMAT NAME: $ABC          LENGTH:      6  NUMBER OF VALUES:      3          |  
|  MIN LENGTH:      1  MAX LENGTH:  40  DEFAULT LENGTH  6  FUZZ:              0          |  
|-----  
| START          | END          | LABEL (VER. V7|V8  10MAR2008:08:20:27) |  
|-----+-----+-----  
| F              | F              | female                                |  
| M              | M              | male                                  |  
| **OTHER**     | **OTHER**     | unkn                                  |  
-----
```

type=I char → numb

INFORMAT NAME: @ABC				LENGTH: 1	NUMBER OF VALUES: 3
MIN LENGTH: 1	MAX LENGTH: 40	DEFAULT LENGTH 1	FUZZ: 0		
START	END	INVALUE(VER. V7 V8	10MAR2008:08:20:27)		
F	F				2
M	M				1
OTHER	**OTHER**	_ERROR_			

type=J char → char

sav4

```
-----  
|      INFORMAT NAME: @$ABC      LENGTH:      5      NUMBER OF VALUES:      3      |  
|  MIN LENGTH:      1  MAX LENGTH:  40  DEFAULT LENGTH  5  FUZZ:      0      |  
|-----  
| START          | END          | INVALUE(VER. V7|V8  10MAR2008:08:20:27) |  
|-----+-----+-----  
| F              | F              | woman                                     |  
| M              | M              | man                                       |  
| **OTHER**     | **OTHER**     | _ERROR_                                  |  
-----
```

cntlout cntlout shows 4 types (note flush)

blst

FMTNAME	TYPE	START	LABEL	HLO
ABC	N	10	men	
		20	women	
		OTHER	else	0
ABC	C	F	female	
		M	male	
		OTHER	unkn	0
ABC	I	F	2	
		M	1	
		OTHER	_ERROR_	0
ABC	J	F	woman	
		M	man	
		OTHER	_ERROR_	0

proc format creates a catalog

```
> proc catalog  cat=work.formats; contents; quit;
```

proc format creates a catalog

clst

Contents of Catalog WORK.FORMATS

#	Name	Type	Create Date	Modified Date	Description
1	ABC	FORMAT	12MAR2008:08:26:36	12MAR2008:08:26:36	
2	ABC	FORMATC	12MAR2008:08:26:36	12MAR2008:08:26:36	
3	ABC	INFMT	12MAR2008:08:26:36	12MAR2008:08:26:36	
4	ABC	INFMTC	12MAR2008:08:26:36	12MAR2008:08:26:36	

format *displays* char,numb iaw format

```
> proc print data=d1;
```

```
> format mf $abc. cod1 abc.; run;
```

NOTE: There were 6 observations read from the data set WORK.D1.

format *displays* char,numb iaw format

Obs	mf	cod1	name	mforig	cod1orig
1	male	men	tom	M	10
2	male	men	dick	M	10
3	male	men	harry	M	10
4	female	women	jane	F	20
5	female	women	sue	F	20
6	female	women	barb	F	20

INPUT *converts* char → numb|char

```
> data d2;  
> set d1 (drop=mforig cod1orig);  
> mf2numb=input (mf, abc.); *- char to numb -;  
> mf2char=input (mf,$abc.); *- char to char -;  
> run;
```

NOTE: There were 6 observations read from the data set WORK.D1.

NOTE: The data set WORK.D2 has 6 observations and 5 variables.

```
> proc print data=d2; run;
```

NOTE: There were 6 observations read from the data set WORK.D2.

INPUT *converts* char → numb|char

Obs	mf	cod1	name	mf2numb	mf2char
1	M	10	tom	1	man
2	M	10	dick	1	man
3	M	10	harry	1	man
4	F	20	jane	2	woman
5	F	20	sue	2	woman
6	F	20	barb	2	woman

PUT *converts* numb|char →char

```
> data d2;  
> set d1 (drop=mforig cod1orig);  
> mf2char = put ( mf,$abc.); *- char to char -;  
> cod2chr = put (cod1, abc.); *- numb to char -;  
> run;
```

NOTE: There were 6 observations read from the data set WORK.D1.

NOTE: The data set WORK.D2 has 6 observations and 5 variables.

```
> proc print data=d2; run;
```

NOTE: There were 6 observations read from the data set WORK.D2.

PUT *converts* numb|char →char

Obs	mf	cod1	name	mf2char	cod2chr
1	M	10	tom	male	men
2	M	10	dick	male	men
3	M	10	harry	male	men
4	F	20	jane	female	women
5	F	20	sue	female	women
6	F	20	barb	female	women

PROC FORMAT

1. beginning basics

→ power practices

3. where warning

an INFORMAT can subset large tables efficiently, aka, table look-up

Advantages:

far, far better than IN (hardcoded list of zips)

better than merge | join for very large tables

since it obviates (tedious) sorts or index creation & usage

simpler than hash table

Method:

cntlin= option creates an informat from a data set

INPUT function *looks up* each candidate zip (large table)

to determine whether it is IN list of desired zips (smaller table)

make a large table

```
> proc surveysselect n=10000 seed=12345 method=urs
>   data=sashelp.zipcode (keep=zip city)
>   out=d1 (keep=numberhits zip city); run;
```

NOTE: The data set WORK.D1 has 8886 observations and 3 variables.

```
> *- make many zips (zips may repeat) not unique -;
```

```
> data zipsmany;
```

```
>   set d1; drop numberhits i;
```

```
>   do i=1 to numberhits; output; end; run;
```

NOTE: There were 8886 observations read from the data set WORK.D1.

NOTE: The data set WORK.ZIPSMANY has 10000 observations and 2 variables.

```
> data d2;                               big_nobs=n1; output; stop;
```

```
>   set zipsmany (drop=zip city) nobs=n1; run;
```

NOTE: The data set WORK.D2 has 1 observations and 1 variables.

```
> proc print data=d2; run;
```

NOTE: There were 1 observations read from the data set WORK.D2.

make a large table

Obs	big_nobs
1	10000

Doug, open the large table

make a lookup table

```
> *-      a few zips I want from many zips table -;
```

```
> data d1;
```

```
> set sashelp.zipcode (keep=zip);
```

```
>      rand1=ranuni(123);
```

```
> if (rand1<0.1); drop rand1; run;
```

NOTE: There were 41988 observations read from the data set SASHELP.ZIPCODE.

NOTE: The data set WORK.D1 has 4135 observations and 1 variables.

```
> proc sort data=d1 out=zipsafew nodupkey; by zip; run;
```

NOTE: There were 4135 observations read from the data set WORK.D1.

NOTE: 0 observations with duplicate key values were deleted.

NOTE: The data set WORK.ZIPSAFEW has 4135 observations and 1 variables.

```
> data d2;          few_nobs=n1; output; stop;
```

```
> set zipsafew (drop=zip)      nobs=n1; run;
```

NOTE: The data set WORK.D2 has 1 observations and 1 variables.

```
> proc print data=d2; run;
```

NOTE: There were 1 observations read from the data set WORK.D2.

make a lookup table

Obs	few_nobs
1	4135

the start var in cntlIN (lookup) table is a primary key (unique)
whereas, the corresponding var in the data (bigger) table is foreign
(may have repeating values)

Doug, open the look-up (smaller) table

sql join

```
> proc sql;
> create table ij1 as select a.zip, city
> from zipsmany a INNER
> JOIN zipsafew b on a.zip=b.zip;
NOTE: Table WORK.IJ1 created, with 964 rows and 2 columns.
> quit;
> data d2;                                sql_nobs=n1; output; stop;
> set ij1 (drop=zip city)                nobs=n1; run;
NOTE: The data set WORK.D2 has 1 observations and 1 variables.
> proc print data=d2; run;
NOTE: There were 1 observations read from the data set WORK.D2.
```

sql join

Obs	sql_nobs
1	964

make informat

```
> data f1;
> length hlo      $1 label 3 type $1  fmtname $3      start $5;
> retain hlo '20'x label 1 type 'I'  fmtname 'abc';
> keep  hlo      label  type      fmtname      start  ;
> if e1 then do; hlo='0'; label=0; OUTPUT; end;
> set zipsafew (keep=zip) end=e1;  drop  zip;
> start=put      (zip,  z5.);  OUTPUT; run;
NOTE: There were 4135 observations read from the data set WORK.ZIPSAFEW.
NOTE: The data set WORK.F1 has 4136 observations and 5 variables.
> proc format
> cntlin =f1  lib=work
> cntlout=o1 (keep=fmtname start label hlo type);
NOTE: Informat ABC has been output.
> run;
NOTE: The data set WORK.O1 has 4136 observations and 5 variables.
NOTE: There were 4136 observations read from the data set WORK.F1.
> data f2;  set f1; if not (hlo='20'x) or _n_<15; run;
NOTE: There were 4136 observations read from the data set WORK.F1.
NOTE: The data set WORK.F2 has 15 observations and 5 variables.
> proc print data=f2 u; run;
NOTE: There were 15 observations read from the data set WORK.F2.
```

make informat

Obs	hlo	label	type	fmtname	start
1		1	I	abc	00650
2		1	I	abc	00670
3		1	I	abc	00687
4		1	I	abc	00715
5		1	I	abc	00716
6		1	I	abc	00729
7		1	I	abc	00733
8		1	I	abc	00740
9		1	I	abc	00742
10		1	I	abc	00751
11		1	I	abc	00802
12		1	I	abc	00823
13		1	I	abc	00928
14		1	I	abc	00930
15	0	0	I	abc	

informat look up

```
> data ij2;
> set zipsmany;
> if input( put (zip,z5.),abc5.); run;
NOTE: There were 10000 observations read from the data set WORK.ZIPSMANY.
NOTE: The data set WORK.IJ2 has 964 observations and 2 variables.
> data d2;                               inp_nobs=n1; output; stop;
> set ij2 (drop=zip)                      nobs=n1; run;
NOTE: The data set WORK.D2 has 1 observations and 2 variables.
> proc print data=d2; run;
NOTE: There were 1 observations read from the data set WORK.D2.
```

informat look up

elst

Obs	inp_nobs	CITY
1	964	

confirm result

```
> proc sort data=ij1 out=ij1s;                by zip; run;
NOTE: There were 964 observations read from the data set WORK.IJ1.
NOTE: The data set WORK.IJ1S has 964 observations and 2 variables.
> proc sort data=ij2                out=ij2s; by zip; run;
NOTE: There were 964 observations read from the data set WORK.IJ2.
NOTE: The data set WORK.IJ2S has 964 observations and 2 variables.
> proc compare data=ij1s comp=ij2s
> noprint    out=ij3;    var zip; run;
NOTE: There were 964 observations read from the data set WORK.IJ1S.
NOTE: There were 964 observations read from the data set WORK.IJ2S.
NOTE: The data set WORK.IJ3 has 964 observations and 3 variables.
> proc print data=ij3 (obs=6) u; run;
NOTE: There were 6 observations read from the data set WORK.IJ3.
> proc means data=ij3 sum; var zip; run;
NOTE: There were 964 observations read from the data set WORK.IJ3.
```

confirm result

Obs	_TYPE_	_OBS_	ZIP
1	DIF	1	00000
2	DIF	2	00000
3	DIF	3	00000
4	DIF	4	00000
5	DIF	5	00000
6	DIF	6	00000

The MEANS Procedure

Analysis Variable : ZIP The 5-digit ZIP Code

Sum

0

contents of CNTLIN

The CONTENTS Procedure

Data Set Name	WORK.F1	Observations	4136
Member Type	DATA	Variables	5

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	FMTNAME	Char	3
5	HLO	Char	1
3	LABEL	Num	3
2	START	Char	5
4	TYPE	Char	1

an informat (type=I) converts char → numb, so

start is char

label is numeric

conforming to this practice is good discipline

contents of CNTLOUT

The CONTENTS Procedure

Data Set Name	WORK.01	Observations	4136
Member Type	DATA	Variables	5

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Label
1	FMTNAME	Char	32	Format name
5	HLO	Char	11	Additional information
3	LABEL	Char	40	Format value label
2	START	Char	9	Starting value for format
4	TYPE	Char	1	Type of format

however, proc format stores all vars as char

“type” controls how format uses start & label

PROC FORMAT

1. beginning basics

2. power practices

→ where warning

data for warning

```
> data d1;  
> length chr $2;  
> input chr @@;  
> datalines;
```

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.

NOTE: The data set WORK.D1 has 12 observations and 1 variables.

```
> ; run;  
> proc print data=d1; run;
```

NOTE: There were 12 observations read from the data set WORK.D1.

data for warning

Obs	chr
1	A
2	Aa
3	Ab
4	az
5	B
6	Ba
7	Bb
8	bz
9	C
10	Ca
11	Cb
12	cz

format for warning

```
> proc format lib=work page
>           cntlout=c1 (keep=fmtname type hlo start end label);
>   invalue
>   abc A=1 B=2
>       other=0;
NOTE: Informat ABC has been output.
>   run;
NOTE: The data set WORK.C1 has 3 observations and 6 variables.
> proc print   data=c1 u; run;
NOTE: There were 3 observations read from the data set WORK.C1.
```

format for warning

```

-----
|      INFORMAT NAME: @ABC      LENGTH:      1  NUMBER OF VALUES:      3      |
|  MIN LENGTH:      1  MAX LENGTH:  40  DEFAULT LENGTH  1  FUZZ:      0      |
|-----|
|START          |END          |INVALUE(VER. V7|V8  12MAR2008:08:29:59)|
|-----+-----+-----|
|A              |A              |                                         |1|
|B              |B              |                                         |2|
|**OTHER**     |**OTHER**     |                                         |0|
-----

```

Obs	FMTNAME	START	END	LABEL	TYPE	HLO
1	ABC	A	A	1	I	
2	ABC	B	B	2	I	
3	ABC	**OTHER**	**OTHER**	0	I	0

informat length usage

```
> data f1;  
> set d1;  
>     val=input( chr,abc1.); *- NOTE: suffix 1. -;  
> run;
```

NOTE: There were 12 observations read from the data set WORK.D1.

NOTE: The data set WORK.F1 has 12 observations and 2 variables.

```
> proc print data=f1; run;
```

NOTE: There were 12 observations read from the data set WORK.F1.

informat length usage

Obs	chr	val
1	A	1
2	Aa	1
3	Ab	1
4	az	0
5	B	2
6	Ba	2
7	Bb	2
8	bz	0
9	C	0
10	Ca	0
11	Cb	0
12	cz	0

informat length subsets correctly

```
> data f2;  
> set d1;  
>     val=input( chr,abc1.); *- NOTE: suffix 1. -;  
> if val; run;
```

NOTE: There were 12 observations read from the data set WORK.D1.

NOTE: The data set WORK.F2 has 6 observations and 2 variables.

```
> proc print data=f2; run;
```

NOTE: There were 6 observations read from the data set WORK.F2.

informat length subsets correctly

Obs	chr	val
1	A	1
2	Aa	1
3	Ab	1
4	B	2
5	Ba	2
6	Bb	2

subset *fails* using WHERE

```
> data f3;  
> set d1;  
> where input( chr,abc1.); *- NOTE: suffix 1. -;  
> run;
```

```
NOTE: There were 2 observations read from the data set WORK.D1.  
      WHERE INPUT(chr, ABC1.);
```

```
NOTE: The data set WORK.F3 has 2 observations and 1 variables.
```

```
> proc print data=f3; run;
```

```
NOTE: There were 2 observations read from the data set WORK.F3.
```

subset *fails* using WHERE

elst

Obs	chr
1	A
2	B

SUBSTR *fixes* problem

```
> data f4;  
> set d1;  
> where input( substr(chr,1,1),abc1.); run;
```

NOTE: There were 6 observations read from the data set WORK.D1.

```
WHERE INPUT(SUBSTR(chr, 1, 1), ABC1.);
```

NOTE: The data set WORK.F4 has 6 observations and 1 variables.

```
> proc print data=f4; run;
```

NOTE: There were 6 observations read from the data set WORK.F4.

SUBSTR *fixes* problem

Obs	chr
1	A
2	Aa
3	Ab
4	B
5	Ba
6	Bb

PROC FORMAT

1. beginning basics

→ power practices – time check

3. where warning

use formats dynamically, at execution time, not compile time

function: inputN – char → numb

function: inputC – char → char

function: putN – numb → char

function: putC – char → char

Due to probable time concerns, I'll only discuss inputN.

data problem – scale by question type (a|b)

```
> proc print data=d2 (obs=17) u;  
> var person quest ab o1-o5; run;
```

NOTE: There were 17 observations read from the data set WORK.D2.

data problem – scale by question type (a|b)

Obs	person	quest	ab	o1	o2	o3	o4	o5
1	p01	q01	b	.	1	.	.	.
2	p02	q01	b	1
3	p03	q01	b	1
4	p04	q01	b	.	1	.	.	.
5	p05	q01	b	.	1	.	.	.
6	p06	q01	b	.	.	.	1	.
7	p01	q02	a	1
8	p02	q02	a	1
9	p03	q02	a	.	.	.	1	.
10	p04	q02	a	.	.	1	.	.
11	p05	q02	a	1
12	p06	q02	a	.	1	.	.	.
13	p01	q03	b	.	.	1	.	.
14	p02	q03	b	.	.	1	.	.
15	p03	q03	b	1
16	p04	q03	b	1
17	p05	q03	b	1

data transposed – long is better than wide

```
> proc print data=d1 (drop=filled obs=17) u; run;
```

NOTE: There were 17 observations read from the data set WORK.D1.

data transposed – long is better than wide

Obs	quest	ab	person	col
1	q01	b	p01	2
2	q01	b	p02	1
3	q01	b	p03	5
4	q01	b	p04	2
5	q01	b	p05	2
6	q01	b	p06	4
7	q02	a	p01	1
8	q02	a	p02	1
9	q02	a	p03	4
10	q02	a	p04	3
11	q02	a	p05	5
12	q02	a	p06	2
13	q03	b	p01	3
14	q03	b	p02	3
15	q03	b	p03	5
16	q03	b	p04	1
17	q03	b	p05	5

informat for conversion

```
> proc format lib=work
> cntlout=f1 (keep=fmtname type hlo start end label);
> invalue aaa
> '1'-'5'=_same_ other=_error_;
NOTE: Informat AAA has been output.
> invalue bbb
> '1'=5 '2'=4 '3'=3 '4'=2 '5'=1 other=_error_;
NOTE: Informat BBB has been output.
> run;
NOTE: The data set WORK.F1 has 8 observations and 6 variables.
> proc print data=f1;
> by fmtname; id fmtname; pageby fmtname; run;
NOTE: There were 8 observations read from the data set WORK.F1.
```

informat for conversion

FMTNAME	START	END	LABEL	TYPE	HLO
AAA	1	5	_SAME_	I	
	OTHER	**OTHER**	_ERROR_	I	0
FMTNAME	START	END	LABEL	TYPE	HLO
BBB	1	1	5	I	
	2	2	4	I	
	3	3	3	I	
	4	4	2	I	
	5	5	1	I	
	OTHER	**OTHER**	_ERROR_	I	0

inputN performs dynamic lookup

```
> data d2;
> set d1; length tmp1 $5;          drop filled;
> tmp1=cat (repeat (ab,2),'1. ');
> val = inputn (put(col,1.),tmp1); *drop tmp1;
NOTE: There were 30 observations read from the data set WORK.D1.
NOTE: The data set WORK.D2 has 30 observations and 6 variables.
> proc print data=d2 (obs=17) u; run;
NOTE: There were 17 observations read from the data set WORK.D2.
```

inputN performs dynamic lookup

Obs	quest	ab	person	col	tmp1	val
1	q01	b	p01	2	bbb1.	4
2	q01	b	p02	1	bbb1.	5
3	q01	b	p03	5	bbb1.	1
4	q01	b	p04	2	bbb1.	4
5	q01	b	p05	2	bbb1.	4
6	q01	b	p06	4	bbb1.	2
7	q02	a	p01	1	aaa1.	1
8	q02	a	p02	1	aaa1.	1
9	q02	a	p03	4	aaa1.	4
10	q02	a	p04	3	aaa1.	3
11	q02	a	p05	5	aaa1.	5
12	q02	a	p06	2	aaa1.	2
13	q03	b	p01	3	bbb1.	3
14	q03	b	p02	3	bbb1.	3
15	q03	b	p03	5	bbb1.	1
16	q03	b	p04	1	bbb1.	5
17	q03	b	p05	5	bbb1.	1

I'm interested in understanding SAS Enterprise Business Intelligence software.
If you use or understand this product, I'd appreciate your help.
douglas.mcallaster@tma.osd.mil or 703-681-3623

typesetting by L^AT_EX with many thanks to ...

Donald Knuth & Leslie Lamport - developers of T_EX & L^AT_EX

Christian Schenk & Thomas Feuerstack - developers of MikT_EX & proT_EXt

Han The Thanh – pdfL^AT_EX (& M. Schroder, H. Henkel, & H. Hagen)

Mark Wick – dvipdfm dvi → pdf conversion software

Karl Berry - president (longtime) of T_EX User Group (TUG)

T_EXLive DVD & CD development & distribution team

Helmut Kopka & Patrick Daly – A Guide to L^AT_EX, 3rd edition

F. Mittlebach, M. Goossens, et al – The L^AT_EX Companion, 2nd edition

package developers (to name a few)

H. Umeki – geometry

R. Schopf – verbatim

S. Rahtz – textcomp