SAS® MACROS: Offer the Best Dating Service

Adel Fahmy, Symbiance Inc., Princeton, NJ

ABSTRACT

Dates occur in all database files that programmers face on a daily basis. Different database designers have adopted a variety of different methods for entering the dates. Some methods create 3 separate fields for Day, Month & Year, and some create only one field that contains the whole date. In either case, the field type may be Numeric or Character. The different methods are usually adopted for the accuracy of data entry or to collect as much information as possible for Partial Dates.

We usually end up with too many macros; each handles one type of data entry date. Those macros are constantly modified to fit in each new project, which defeats the idea of generic macros. The macro presented here handles most of the commonly used date fields producing a Full Numeric Date (used for calculations) and a Full Character Date (used for reporting), with the hope that we have solved most, if not all, of your dating issues.

KEYWORDS

Date, Partial Date, Numeric Date, Character Date.

AUDIENCE

Database Professionals, Programmers & Statisticians with limited or advanced SAS experience.

FORMATS OF DATES

Single-Field Date: Where the date is entered as one continuous string that includes Day, Month, and Year. The order of the field parts depends on country convention, company standards, or the database designer. The most common formats are: mmddyyyy, ddmmyyyy, or yyyyymmdd

Multi-Field Date: Where the date is entered as 3 separate fields that contain Day, Month, and Year. The most common is: dd, mm, and yyyy

THE 2-DIGIT YEAR ISSUE

After the Y2K issues have been encountered, almost all companies use 4-digits (yyyy) to represent the year. Very few are still using the 2-digit year (yy) in their new projects. For older projects, you will definitely encounter the 2-digit year. The 2-digit year issue is beyond the scope of this paper. Since it is rarely used at present, it has not been considered at this time to avoid more complexity of the macro.

TYPES OF DATES

Numeric Dates: Where the single field that holds the full date or the multiple fields that hold the date parts are all numeric. The single date field, although numeric, is not necessarily a valid SAS date that could be used for calculations of a duration period. This usually occurs in the 3 separate fields case, with the lack of online validation (at data entry time). The online validation of the 3-field inter-relationship is usually skipped to avoid slowing down the professional data entry personnel.

Character Dates: Where the single field that holds the full date or the multiple fields that hold the date parts are all characters. The character fields are usually used to collect partial dates. Single date field is not necessarily a valid SAS date that could be used for calculations of a duration period. For character dates, anything goes: company standards, boss standards, leading or no leading zeros, entering missing date parts as 00, --, **, ##, MM, etc. With data entry errors in addition, you will need tons of IF conditions in order to cover the possible permutations of what might have been entered.

NUMERIC VERSUS CHARACTER DATES

It is important here to indicate that in statistical tabulations, the Numeric date should be listed because this is one that has been used for calculations and classification of the population. As an example:

Treatment Period = Visit Date – First Dose Date.

On the other hand, Character dates should be presented “as is” in regular reports or listings. Most data listings are used by data management personnel to validate and correct database errors. It creates more confusion if the listed date is different from the one entered.
EUROPEAN DATE FORMATS

European date format is usually in the form ddmmyyyy with or without slash/dash separators between the day, month and year. The macro handles those dates in a similar way to the American format mmddyyyy, by capturing the day, month and year separately, then constructing the unified format ddmmyyyy. It should be noted here that many Asian and African countries use the European format. Since this issue will be faced in global projects, it will be considered by the macro.

PARTIAL DATES

Partial Dates are particularly important in the medical and pharmaceutical industries. For example, it is important to collect Date of Previous Surgery, even if the information known at present is the month and year or only the year of the event.

In a numeric full-date field, formatted as a SAS date, you cannot enter partial dates. In such case, partial dates will be simply missing. 3-field numeric dates allow entering partial dates. Character dates allow entering partial dates. Although you are expecting date values in the fields, they allow entering anything including letters or special characters. This is the programmer’s nightmare of character fields. Too many validation checks have to be written before utilizing the characters; are they purely numeric or do they include alphabetic as well?

STANDARDIZATION

Leading zeros are very important to use, particularly in 1-part dates. It is a good practice to also use them in 3-part dates. For Partial Dates, specific and unified conventions have to be adopted in entering missing date parts. Whatever the convention (for example blank, 00, --, **, #), the user is given control of how to represent it in printed output.

WHERE HAVE THE LEADING ZEROS GONE?

In Character dates, the leading zeros will be there all the time since they are like any other alphabetic character. But in Numeric dates, although you may input your data with leading zeros, SAS stores them without (example 01012004 is stored as 1012004). This crucial point has to be considered when you “SUBSTR”ing the date. The macro will take this into account during the identification process of the Day, Month and Year. Examples:

- 7-char date 1112004
  - If INFMT = mmddyyyy then the date is 11JAN2004.
  - If INFMT = ddmmyyyy then the date is 01NOV2004.
- 5-char date 12004
  - Whether INFMT = mmddyyyy or INFMT = ddmmyyyy, the date is **JAN2004.

PURPOSE OF PROGRAM

To transfer data entry dates in different formats, numeric or character, full or partial, into valid numeric SAS dates (for calculation purposes) and character dates (for reporting purposes).

- Date entered in 3 Parts DD, MM & YYYY (Character or Numeric)
- Date entered in 1 Part DDMMYYYY, MMDDYYYY, or YYYYMMDD (Character or Numeric)
- The resulting Full Dates (Character and SAS Numeric) DTvarC and DTvarN will be in the form: DDMMYYYY (default), DDMMYYYY, MMDDYYYY, or YYYYMMDD.

OTHER DATE STYLES

Valid Full Numeric Dates, although the macro can easily handle them, do not need to use this or any other macro. Simply output using SAS provided Date Formats. For date variables that include separators, such as dd-mm-yyyy, or mm/dd/yyyy, split the date into 3 character parts using the SUBSTR or SCAN function, then use the macro to handle them.

MACRO COMPILING

In order to compile the macro, use the following statements before the macro definition. This will create the catalog file SASMACR.SAS7BCAT (under Microsoft Windows). In order to use the compiled macro, the same statements are used to define where the compiled macro is located. Using the compiled macro preserves the generic code and eliminates the need for using %INC to include the source code. The macro library MAC needs to be defined only ONCE in a session. The macro library cannot be redefined or cleared until the end of the SAS session.

```sas
%LET mac = %STR (c:|_res_ea_r\ch\dat e\mac ro\s) ;
LIBNAME mac V8 "&mac =" ;
OPTIONS MSTORED SASMSTORE = mac ;
```
DESCRIPTION

The process is automated through one and only one generic macro. It will create two new variables, DtvarN and DtvarC.

(0) THE CALLING PROGRAM

- To call the macro, assume that a dataset MDY3C has Input Date in 3 Separate Character Fields mm, dd & yyyy. To output the date (Character and Numeric) in the form ddmmyyyy with missing values filled by (*), use the following macro call. [ More examples including sample data are shown in Appendix 1 ].

```sas
%date ( indata = mdy3c , infmt  = mm  dd   yyyy ,
   invar  = tstm tstd tsty ,
   outlbl = Test Date ,
   outfmt = ddmmmyyyy ,
   outvar = test ,
   blank  = * ) ;
```

```sas
PROC PRINT NOOBS DATA = mdy3c ; RUN;
```

(1) DATE Macro

DEFINITION:
- The macro has 7 parameters. The STORE option will compile the macro definition in a permanent catalog for later use.

```sas
%MACRO date ( indata = , infmt  =           , invar  = ,
   outlbl = , outfmt = ddmmmyyyy , outvar = , blank = * )
   / STORE DES = ‘Transfer Date into Num & Char Dates’ ;
```

MACRO INPUT PARAMETERS:
- **INDATA** input dataset to be processed.
- **INFMT** format of the input date including the order of day, month & year. Each character representing the day, month & year should be represented by d, m & y, respectively. For dates in 3 separate fields: dd mm yyyy. For dates in 1 field: ddmmyyyy, mmddyyyy, or yyyymmdd.
- **INVAR** input date variable name(s), in the same order as the INFMT. For dates in 3 separate fields: Date-parts Variable Names vard varm vary in the Exact INFMT Order dd mm yyyy. For dates in 1 field: Date Variable Name vardt.
- **OUTLBL** output date variable label, excluding quotes.
- **OUTFMT** output date format for OUTVAR ddmmyyyy (Default), ddmmyyyy, mmddyyyy, or yyyymmdd.
- **OUTVAR** output date variable name, the main body of the name. Example: test, visit, exam. Will be auto added: a prefix "DT" and a suffix "C" or "N". DTOutvarN Numeric date variable name, using outvar with prefix DT and suffix N. DTOutvarC Character date variable name, using outvar with prefix DT and suffix C.
- **BLANK** a filler character for blanks in character date without quotes (space, -, *, #). Default is Asterisk (*). In other words, how do you want the missing values to be represented in OUTVAR? Note: It is not recommended to use BLANK=- in case of OUTFMT=yyyymmdd, since there will be dash separators between the year, month & day.

TYPE OF INPUT VARIABLE(s):
- Get the Type of input date field(s) INVAR (Numeric or Character), so the user does not have to specify it. Put the type in a macro variable INTYP. The macro will then execute accordingly.

```sas
PROC CONTENTS NOPRINT DATA = &indata OUT = type ( KEEP = name type ) ; RUN ;
```
DATA type;
SET type END = eof;

IF INDEX ( UPCASE ( "&invar" ) , UPCASE ( COMPRESS ( name ) ) ) = 0 THEN DELETE;
IF type = 1 THEN typ = "N";
ELSE IF type = 2 THEN typ = "C";
ELSE                              typ = " ";
IF eof THEN CALL SYMPUT ( "intyp" , typ );
DROP type;
RUN;

PROCESSING INPUT DATA:
• Consider the current year is the Maximum valid year and 99 years earlier to be the Minimum valid year. Define 3 numeric variables ddn, mmn & yyn to save the numeric parts of the date and 3 character variables ddc, mmc & yyc to save the corresponding character parts. The local macro variables dd, mm & yy will store the name of the corresponding variable. Now, get the Day, Month & Year parts of the date. Validate the Parts before constructing the Full Numeric Date. Assume that missing values are represented by: m = missing, u = unknown, i = illegible.

%LOCAL dd mm yy ;

DATA &indata ;
SET &indata ;
MISSING m u i ;
Maxyr = YEAR ( TODAY ( ) ) ;
Minyr = maxyr - 99 ;
Num = '0123456789' ;

SPLIT INPUT VARIABLES:
• Split INVAR into Parts (dd, mm, yy) based on INFMT. For INFMT=dd mm yyyy, make sure that the 3 parts are being separated with at least 2 spaces. For INFMT=ddmmyyyy, based on the string length the macro will satisfy first the year, then month, then day. To simplify the idea, all dates will have a string length of 8 characters. 7-Character dates will be padded with 1 left space, and 5-Characters dates will be padded with 3 left spaces, then all will be treated similar to 8-character dates. Non-missing dates of length 3 characters or less result in an error.

%IF %UPCASE ( &infmt ) = %STR (DD MM YYYY) &
  ( %LENGTH ( &invar ) - %LENGTH ( %SYSFUNC ( COMPRESS ( &invar ) ) ) ) >= 2
%THEN  %DO ;
%LET dd = %SCAN ( &invar , 1 ) ;
%LET mm = %SCAN ( &invar , 2 ) ;
%LET yy = %SCAN ( &invar , 3 ) ;
%END ;
%* ________________________________________________________________________________ ;

%ELSE  %IF %UPCASE ( &infmt ) = %STR (MM DD YYYY) &
  ( %LENGTH ( &invar ) - %LENGTH ( %SYSFUNC ( COMPRESS ( &invar ) ) ) ) >= 2
%THEN  %DO ;
%LET mm = %SCAN ( &invar , 1 ) ;
%LET dd = %SCAN ( &invar , 2 ) ;
%LET yy = %SCAN ( &invar , 3 ) ;
%END ;
%* ________________________________________________________________________________ ;

%ELSE  %IF %UPCASE ( &infmt ) = %STR (DDMMYYYY)
%THEN  %DO ;
%LET dd = dd ;  %LET mm = mm ;  %LET yy = yy ;
IF VERIFY ( COMPRESS ( &invar ) , num ) = 0

THEN DO;
  IF LENGTH ( COMPRESS (&invar) ) = 8
  THEN DO;
    dd = SUBSTR ( COMPRESS (&invar), 1, 2 );
    mm = SUBSTR ( COMPRESS (&invar), 3, 2 );
    yy = SUBSTR ( COMPRESS (&invar), 5, 4 );
  END;
  ELSE IF LENGTH ( COMPRESS (&invar) ) = 7
  THEN DO;
    dd = SUBSTR ( COMPRESS (&invar), 1, 1 );
    mm = SUBSTR ( COMPRESS (&invar), 2, 2 );
    yy = SUBSTR ( COMPRESS (&invar), 4, 4 );
  END;
  ELSE IF LENGTH ( COMPRESS (&invar) ) = 6
  THEN DO;
    mm = SUBSTR ( COMPRESS (&invar), 1, 2 );
    yy = SUBSTR ( COMPRESS (&invar), 3, 4 );
    dd = '  ';
  END;
  ELSE IF LENGTH ( COMPRESS (&invar) ) = 5
  THEN DO;
    mm = SUBSTR ( COMPRESS (&invar), 1, 1 );
    yy = SUBSTR ( COMPRESS (&invar), 2, 4 );
    dd = '  ';
  END;
  ELSE IF LENGTH ( COMPRESS (&invar) ) = 4
  THEN DO;
    yy = SUBSTR ( COMPRESS (&invar), 1, 4 );
    mm = '  ';
    dd = '  ';
  END;
END;  
%END;
%* ________________________________________________________________________________ ;
%ELSE %IF %UPCASE ( &infmt ) = %STR (MMDDYYYY)
%THEN %DO:
  /* [ The process is very similar to the previous for &INFMT=DDMMYYYY ] */
%END;
%* ________________________________________________________________________________ ;
%ELSE %F %UPCASE ( &infmt ) = %STR (YYYYMMDD)
%THEN %DO:
  /* [ The process is very similar to the previous for &INFMT=DDMMYYYY ] */
%END;
%* ________________________________________________________________________________ ;
NUMERIC DATE PARTS:
• Transfer Date Parts into Numeric if they were provided as Chars. Put Leading Zeros in front of Single Digit Day or Month.

```
FORMAT ddn mmn yyn z2. yyn z4. ;
% F %UPCASE ( &intyp ) = N
%THEN %DO ;
  IF &dd ne . THEN ddn = PUT ( &dd*1 , z2. ) ;
  IF &mm ne . THEN mmn = PUT ( &mm*1 , z2. ) ;
  IF &yy ne . THEN yyn = PUT ( &yy*1 , z4. ) ;
%END;
%* ________________________________________________________________________________ ;
%ELSE % F %UPCASE ( &intyp ) = C
%THEN %DO ;
  IF &dd ne ' ' THEN ddn = PUT ( &dd*1 , z2. ) ;
  IF &mm ne ' ' THEN mmn = PUT ( &mm*1 , z2. ) ;
  IF &yy ne ' ' THEN yyn = PUT ( &yy*1 , z4. ) ;
%END ;
```

CHARACTER DATE PARTS:
• Construct a Character Date from input data as is, replace Blanks with a default filler Asterisk (*).

```
FORMAT ddc mmc yyc $2. $4. ;
% F %UPCASE ( &intyp ) = N
%THEN %DO ;
  IF &dd NE . & VERIFY ( COMPRESS ( &dd ) , num ) NE 0 THEN ddc = LEFT ( PUT ( &dd , 2. ) ) ;
  ELSE IF &dd NE . THEN ddc = LEFT ( PUT ( ddn , z2. ) ) ;
  IF &mm NE . & VERIFY ( COMPRESS ( &mm ) , num ) NE 0 THEN mmc = LEFT ( PUT ( &mm , 2. ) ) ;
  ELSE IF &mm NE . THEN mmc = LEFT ( PUT ( mmn , z2. ) ) ;
  IF &yy NE . & VERIFY ( COMPRESS ( &yy ) , num ) NE 0 THEN yyc = LEFT ( PUT ( &yy , 4. ) ) ;
  ELSE IF &yy NE . THEN yyc = LEFT ( PUT ( yyn , z4. ) ) ;
%END ;
%* ________________________________________________________________________________ ;
%ELSE % F %UPCASE ( &intyp ) = C
%THEN %DO ;
  IF &dd NE ' ' & VERIFY ( COMPRESS ( &dd ) , num ) NE 0 THEN ddc = LEFT ( PUT ( &dd , $2. ) ) ;
  ELSE IF &dd NE ' ' THEN ddc = LEFT ( PUT ( ddn , z2. ) ) ;
  IF &mm NE \no\ & VERIFY ( COMPRESS ( &mm ) , num ) NE 0 THEN mmc = LEFT ( PUT ( &mm , $2. ) ) ;
  ELSE IF &mm NE ' ' THEN mmc = LEFT ( PUT ( mmn , z2. ) ) ;
  IF &yy NE ' ' & VERIFY ( COMPRESS ( &yy ) , num ) NE 0 THEN yyc = LEFT ( PUT ( &yy , $4. ) ) ;
  ELSE IF &yy NE ' ' THEN yyc = LEFT ( PUT ( yyn , z4. ) ) ;
%END ;
```
CHARACTER MONTH:
- Find the Month abbreviation corresponding to the month numeric value.

    FORMAT mmc $3. ;
    SELECT ( mmc ) ;
        WHEN (01) mmmc = 'JAN' ;
        WHEN (02) mmmc = 'FEB' ;
        WHEN (03) mmmc = 'MAR' ;
        WHEN (04) mmmc = 'APR' ;
        WHEN (05) mmmc = 'MAY' ;
        WHEN (06) mmmc = 'JUN' ;
        WHEN (07) mmmc = 'JUL' ;
        WHEN (08) mmmc = 'AUG' ;
        WHEN (09) mmmc = 'SEP' ;
        WHEN (10) mmmc = 'OCT' ;
        WHEN (11) mmmc = 'NOV' ;
        WHEN (12) mmmc = 'DEC' ;
        WHEN (.M) mmmc = 'M  ' ;
        WHEN (.U) mmmc = 'U  ' ;
        WHEN (.I) mmmc = 'I  ' ;
        OTHERWISE mmmc = mmc ;
    END ;

FULL CHARACTER DATE:
- Based on requested output format OUTFMT, add 2 slashes or dashes as necessary. Put the Character Date Parts together in one piece.

    LABEL dt&outvar.c = "&outlbl - Char" ;

    %IF %UPCASE ( &outfmt ) = DDMMYYYY  %THEN %DO ; FORMAT dt&outvar.c $10. ; %END ;
    %ELSE %IF %UPCASE ( &outfmt ) = MMDDYYYY  %THEN %DO ; FORMAT dt&outvar.c $10. ; %END ;
    %ELSE %IF %UPCASE ( &outfmt ) = YYYYMMDD  %THEN %DO ; FORMAT dt&outvar.c $10. ; %END ;
    %ELSE %IF %UPCASE ( &outfmt ) = DDMMMYYYY %THEN %DO ; FORMAT dt&outvar.c $9.  ; %END ;
    %ELSE                                           %DO ; FORMAT dt&outvar.c $9.  ; %END ;
    SELECT ( UPCASE ( "&outfmt" ) ) ;
        WHEN ("DDMMYYYY" ) dt&outvar.c =PUT(ddc,$2.)||'/'||PUT(mmc,$2.)||'/'||PUT(yyc,$4.);
        WHEN ("MMDDYYYY" ) dt&outvar.c =PUT(mmc,$2.)||'/'||PUT(ddc,$2.)||'/'||PUT(yyc,$4.);
        WHEN ("YYYYMMDD" ) dt&outvar.c =PUT(yyc,$4.)||'-'||PUT(mmc,$2.)||'-'||PUT(ddc,$2.);
        WHEN ("DDMMMYYYY") dt&outvar.c =TRIM(PUT(ddc,$2.)||PUT(mmmc,$3.)||PUT(yyc,$4.));
        OTHERW SE          dt&outvar.c =TRIM(PUT(ddc,$2.)||PUT(mmmc,$3.)||PUT(yyc,$4.));
    END ;

FILLING THE BLANKS:
- Add 2 Slashes/Dashes to LOUTFMT, the length of OUTFMT, then check the length of DT&outvar.C. Replace blanks with a user defined filler character BLANK ( space, -, *, #, etc ). Default is the Asterisk (*).

    FORMAT loutfmt 2. ;
    SELECT ( UPCASE ( COMPRESS ( "&outfmt" ) ) ) ;
        WHEN ("MMDDYYYY" ) loutfmt = LENGTH ( COMPRESS ( "&outfmt" ) ) + 2 ;
        WHEN ("DDMMYYYY" ) loutfmt = LENGTH ( COMPRESS ( "&outfmt" ) ) + 2 ;
        WHEN ("YYYYMMDD" ) loutfmt = LENGTH ( COMPRESS ( "&outfmt" ) ) + 2 ;
        WHEN ("DDMMMYYYY") loutfmt = LENGTH ( COMPRESS ( "&outfmt" ) ) ;
        OTHERW SE          loutfmt = LENGTH ( COMPRESS ( "&outfmt" ) ) ;
    END ;
IF LENGTH ( COMPRESS ( dt\&outvar.c ) ) < loutfmt
THEN dt\&outvar.c = TRANSLATE ( dt\&outvar.c , "\&blank", " " ) ;

FULL NUMERIC DATE:
Construct a Numeric Date from Valid INPUT Data after Validating the Parts.

%IF %UPCASE ( &outfmt ) = %STR( DDMMYYYY )  %THEN %DO; %LET outfmt=%STR(ddmmyy10.);%END;
%ELSE %IF %UPCASE ( &outfmt ) = %STR( MMDDYYYY )  %THEN %DO; %LET outfmt=%STR(mmddyy10.);%END;
%ELSE %IF %UPCASE ( &outfmt ) = %STR( YYYYMMDD )  %THEN %DO; %LET outfmt=%STR(yymmdd10.);%END;
%ELSE %IF %UPCASE ( &outfmt ) = %STR( DDMMMYYYY ) %THEN %DO; %LET outfmt=%STR(date9.   );%END;
%ELSE                                             %DO; %LET outfmt=%STR(date9.   );%END;

FORMAT dt\&outvar.n &outfmt ;
LABEL  dt\&outvar.n = "\&outlbl - Num" ;

%IF  %UPCASE ( &infmt ) = %STR (DD MM YYYY) OR
    %UPCASE ( &infmt ) = %STR (MM DD YYYY) OR
    %UPCASE ( &infmt ) = %STR (DDMMYYYY)   OR
    %UPCASE ( &infmt ) = %STR (MMDDYYYY)   OR
    %UPCASE ( &infmt ) = %STR (YYYYMMDD)   OR
THEN %DO ;

IF 1<= LENGTH ( COMPRESS (ddn)) <= 2 &
   1<= LENGTH ( COMPRESS (mmn)) <= 2 &
   1<= LENGTH ( COMPRESS (yyn)) <= 4 &
   VERIFY ( TRIM ( LEFT (ddn)) , num ) = 0 &
   VERIFY ( TRIM ( LEFT (mmn)) , num ) = 0 &
   VERIFY ( TRIM ( LEFT (yyn)) , num ) = 0 &

   01 <= ddn <= 31   &
   01 <= mmn <= 12   &
   minyr <= yyn <= maxyr &

   ( ( mmn in ( 4 6 9 11)   &   1 <= ddn <= 30 ) |
     ( mmn in ( 1 3 5 7 8 10 12) &   1 <= ddn <= 31 ) |
     ( mmn in ( 2)       & ((1 <= ddn <= 28 ) |
       (ddn=29 & MOD ( yyn , 4 ) = 0 ) ) ) )

THEN  dt\&outvar.n = mdy ( mmn , ddn , yyn ) ;
ELSE  dt\&outvar.n = . ;
%END;

DROP num minyr maxyr ddn mm yyn ddc mmn yyc mmnt loutfmt
%IF %UPCASE ( &outfmt ) = %STR (DDMMYYYY) OR
    %UPCASE ( &outfmt ) = %STR (MMDDYYYY) OR
    %UPCASE ( &outfmt ) = %STR (YYYYMMDD)
%THEN %DO ; dd mm yy %END ;

RUN ;

%MEND date ;
CONCLUSION

This macro will transfer dates of different types and formats into full usable Numeric and Character dates. The macro covers most of the commonly used data entry formats. The macro is Generic and does not need to be modified for specific projects. It handles missing information as well as partial dates. There is no longer a need for many different macros to handle dates.

FUTURE WORK

For systems that use standard naming conventions, some of the macro input parameters could be eliminated. By using some of the SAS Functions, those parameters could be automatically identified by the macro, minimizing the user requirements. The macro could be utilized in Edit Checks of dates by comparing the resulting character and numeric dates. If a complete character date was provided but results in no numeric one, then the date should be flagged with an error message.

VALIDATION

The macro and sample calls have been fully tested and validated using SAS Version 8.02 and 6.12 software on Microsoft Windows ® platform.

REFERENCES

• SAS Base, Language Reference, V6, E1, Feb 1996.

AUTHOR

Adel Fahmy is an Independent Consultant. Previously, he worked as Sr. Statistical Programmer and Associate Director of Systems at major pharmaceutical companies and clinical research organizations, and as internal consultant at universities, teaching SAS to faculty and students. Adel has been a SAS user for 20 years. His special achievements include Generic Macros, Edit Checks, Database Design, Menu-Driven Systems, Optimization Techniques and Project Management.

Adel has BS in Mathematics, Graduate Diploma in Systems Design, MS and M.Phil. in Computer Science from Nottingham University, UK.

CONTACT

Your comments and questions are welcomed. The author can be contacted at:
Adel Fahmy, EMS: Adel.Fahmy@att.net Voice: (732) 422 6651

COPYRIGHTS

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. Windows® is a registered trademark of Microsoft Inc. Other brand and product names are registered trademarks or trademarks of their respective companies.