Paper 100-28

MACRO Function with Error Handling to Automatically Generate Global Macro Date Variables.

Anatoly V. Kulinsky, Fleet Credit Card Services, subsidiary of FleetBoston Financial, Horsham, PA

ABSTRACT

Custom macro function **'ANYDATE'** is designed to generate a requested number of global macro variables representing dates. The **INTNX** function is used to produce dates in required format for different number of time intervals going forward or backward. **CALL SYMPUT ROUTINE** uses date values to create macro variables each representing date in specified format. Thus allowing referencing and comparing any date values stored in the variables/fields or embedded into SAS® libname, dataset name or variable name; External directory/file /field name; Database schema/table/column name for both processing and reporting purposes. **%MACRO 'ANYDATE'** was developed to automate data pull from Oracle DB and to create SAS Data Mart from individual datasets. Global macro variables were used to reference date dimension either embedded into table name or using 'date' variables on indexed/partitioned tables with 'WHERE' statement. Further automation is achieved by making it customizable, introducing several arguments allowing to produce various return values -- macro date variables. Since the function is designed as an application -- utility tool for users, elaborate Error Handling was introduced to exit program using %**GOTO** and %**LABEL** statements with explanatory **ERROR** message. This function written for SAS v.8.1 both UNIX and Windows environment. It could be used with minor changes in version 6.x. Intended for audience with intermediate to advanced skills.

INTRODUCTION

Many routine tasks performed in Financial and Pharmaceutical industries depend on date values. Analyses are made over certain period, within either that interval or spanning over a time series for data comparison. In the case of Credit Card Industry, data could be organized as multi-record transactions or summarized on account level in cycles or as of month-end. Most of the original data comes as daily files. Those carry information on customers, transactions such as balance transfers, cash advances, purchases, returns. Most financial data for instance cash and merchandise APR, balances and fees, cost of funds and so on are updated on monthly basis. To further complicate an issue some of the most important information from Credit Bureaus comes on the quarterly basis.

ASSUMPTIONS

1. The user familiar with **Date Formats** in programming languages such as SAS or SQL.
2. The user aware of how the **INTNX** function works and notion of interval boundaries.
3. Value of Macro variables been created defines by the argument Format. Convention – 'Y' defines year; 'Q' defines quarter; 'M' defines month; 'W' defines week and 'D' defines day. Any **Format** that includes 'Date' defines SAS **DATEw. Format.**
4. Number of key-letters like 'Y' or 'M' define specifics of **Format** requested and total length of value of Macro variables. Three 'M' or 'MON' defines **MonYYw Format.**
5. Arguments **Interval** and **Format** should not be mismatched!

Name of the first global macro variable does not have numeric index, whereas all additional does with most recent having '1' , next '2' and so on.

CONCEPT

SAS dates are special numbers stored as integers. Dates before January 1, 1960 are negative integers and after January 1,1960 are positive. However, we are used to different representation of time, no one can easily tell what date is 15796 or –21549. To use them we need to see customary dates. We habitually use formats typically: **DATEw**. , **MMDDYYw**. or **YYMMDDw**. Forms of these formats are consistent and self-explanatory but format length '**w'** -- not always. It is not easy to remember how '03MAR03'd will be written out by MMDDYY5. – as 33103 or 03/31 or 03-03. Most convenient for processing purposes, I think, are YYMMn6. or YYMMDDn8., where 'n' stands for no separator, thus avoiding inconsistency and introduction of any special characters. Placing year first followed by month allows easy sort and comparison of any dates in both numeric and character form. Function **TODAY**() is the source of the default start date unless user needs specific or constant date. Function **INTNX** generates a SAS date value that is shifted by specified number of intervals from the start. **CALL SYMPUT** routine uses date values to create requested number of global macro variables each representing date in specified format.

%MACRO anydate is designed to accommodate a variety of possible factors, build SAS code dynamically based on supplied arguments. Due to such complexity and to help end-users with various skill level led to the utilization of SAS LOG to control job execution with messages. **Error Handling** was introduced. Rationale is to test potential variations of all function's arguments for errors or for their mutual incompatibility and either reassign the value with explanatory message or terminate program execution using %**GOTO** and %**LABEL** statements with detailed descriptive **ERROR** message in the log.

TECHNICAL DETAILS

In the foundation is date function **INTNX**. It generates a SAS date value that is a specified number of intervals from a starting value. Its syntax is:

INTNX ('interval', start-from, increment <, 'alignment'>).

For example:

```
mdate = intnx('month', '31MAR2003'd , -1, ‘m’ );
```

Will assign value 15750 to the variable mdate.

CALL SYMPUT routine uses date values to generate macro variables.

```
   CALL SYMPUT('yymm', put(mdate, yymmdd6.));
```

That will create macro variable yymm with value of 030214.

Function **TODAY**() as the source of the default start date were chosen over more intuitive automatic macro variable &**SYSDATE** due to ambiguous behavior of it under Windows environment.

Macro has seven keyword parameters. Parameter "startdt" should be specified if hardcoded date is desirable whereas the other six have default values. They are: **(1)** "Format" to represent desired date format and to evaluate length for “w” value, **(2)** ”Interval” – time Interval (step) like month or week, **(3)** ”Incremnt” to request number of steps, **(4)** ”Align” – alignment to the beginning, middle or the end of the period and 'b','m','e' can be used as abbreviation., **(5)** ”First” – start point of additional date variables, **(6)** ”Last” – end point.

SOURCE CODE

```
   %MACRO anydate(format=yrmo, interval=month,
   incremnt=0,align=b,first=1,last=6,startdt= );

   %LOCAL n m incremnt directn align first last
   interval format;

   %IF %SYSFUNC(INDEXC(&format,%str(/:\|~#!^*-
   _+.[]{}= ;<>))))>0 %THEN
     %DO;
   %put> WARNING: FORMAT "&FORMAT" is invalid!
   It SHOULD not contain special characters.;
   %put > FORMAT has been converted to "&FORMAT"
   for processing.;
   %let format=%SYSFUNC(compress(&format,
   %str(/:\|~#!^*-_+.[]{}= ;<>)));
   %put > FORMAT has been converted to "&FORMAT"
   for processing.;
    %END;
         /* reassignment and Change Case */
   %GLOBAL &format;
  %LET format=%LOWCASE(%TRIM(%LEFT(&format)));
%LET interval=%LOWCASE(%TRIM(%LEFT(&interval)));

   %IF %length(&format)<4 %THEN
     %DO;
   %put>ERROR: FORMAT "&FORMAT" is too short! *;
   %put>There SHOULD be at least four characters
   like: YYMM or YYYR.*;
   %goto errhandl;
     %END;
   %ELSE %IF %length(&format)>8 %THEN
     %DO;
       ...
   %ELSE %IF %SYSFUNC(INDEXC(&format,dwmqy))=0
   or %SYSFUNC(INDEXC(&interval,dwmqy))=0 %THEN
     %DO;
   %put>ERROR: FORMAT "&FORMAT" or INTERVAL
   "&INTERVAL" are invalid! They SHOULD contain
   YEAR and MONTH or QTR or WEEK or DAY in
   references like: YRMO, MONYYYR, MONYR, YYYR,
   YRQTR, YYYRQTR*;
   %goto errhandl;
     %END;
   ...
   %ELSE %IF %INDEX(&format,date)>0 %THEN
     %DO;
   %IF %INDEX(&format,date9)>0 %THEN %let
   format=date_full;
         %ELSE %IF %INDEX(&format,date)>0 %THEN
   %let format=datesvn;
       ...
   %ELSE
     %DO;
       %put *** FORMAT "&FORMAT" and INTERVAL
   "&INTERVAL" are valid!*;
     %END;
   ...
   %IF %SYSFUNC(INDEXC(&align,bme))=0 %THEN
     %DO;
   %put>ERROR: ALIGNMENT "&align" is invalid!;
   %put>It SHOULD be 'b' or 'm' or 'e' only.*;
   %goto errhandl;
     %END;
    %ELSE
  %DO;
    %put *** ALIGNMENT "&align" is valid!*;
  %END;
...
         /* Start of INTERVAL processing*/

%IF %INDEX(&interval,m)>0 %THEN
   %DO; /* interval=month */
      %IF %INDEX(&format,q)>0 %THEN %DO;
%put>ERROR: INTERVAL "&INTERVAL" and FORMAT
"&FORMAT" are MISMATCHED! FORMAT "&FORMAT"
should not contain QTR reference.*;
%goto errhandl;
%END;
%ELSE %IF %SYSFUNC(INDEXC(&format,dm))>0
%THEN %let interval=month;
...
      /* Start of FORMAT processing */

%IF %INDEX(&format,q)>0 %THEN %let dtfmt=yyq;
 %ELSE %IF %length(&format)<5 %THEN
   %DO;
      %IF %INDEX(&format,date)>0 %THEN
                   %let dtfmt=date;
      %ELSE %IF %SYSFUNC(INDEXC(&format,dw))>0
      %THEN %DO;
%put>ERROR: FORMAT "&FORMAT" is too SHORT!
For DAY FORMAT should contain both YEAR and
MONTH reference as well.;
%GOTO errhandl;
        %END;
      %ELSE %IF %INDEX(&format,m)=0 %THEN
       %DO;
%put>ERROR: SHORT FORMAT "&FORMAT" is
INCORRECT! The FORMAT should contain: MONTH
or QTR or YEAR!*;
 %GOTO errhandl;
   %END;
    %ELSE
    %DO;
    %IF %INDEX(&format,y) > %INDEX(&format,m)
    %THEN %let dtfmt=mmyyn;
    %ELSE %let dtfmt=yymmn;
    %END;
   %END;
 %ELSE
   %DO; /* %IF %length &format > 4 */
...
%IF &interval=year %THEN %let dtfmt=year;

 %IF %INDEX(&incremnt,-) > 0 %THEN
   %DO;
       %let directn=%str(-);
          %put * Direction for INTNX function
    and sign is "&directn".*;
%END;
 %ELSE
   %DO;
       %let directn= ;
            %put * Direction for INTNX
            function and sign is "+".;
%END;

DATA _NULL_ ;

%IF &startdt=%THEN
   %DO;
   * if startdt missing then use sysdate;

sysd = INTNX("&interval", TODAY(), &incremnt,
"&align");
            %put>Start date for INTNX
            function is "&startdt";
%END;
 %ELSE
   %DO;
sysd = INTNX("&interval", input("&startdt",
yymmdd%length(&startdt).), &incremnt,
"&align");
%put>Start date for INTNX function is
"&startdt" and fmt is &dtfmt%length(&format).
with length %length(&startdt) bytes.*;
%END;
%put * Generating Dates for format "&format"
using SAS format &dtfmt%length(&format). with
Alignment "&align". *;

CALL SYMPUT("&format",LOWCASE(TRIM(LEFT
(PUT(sysd,&dtfmt%length(&format).)))));

     %IF %EVAL(&last >= &first) %THEN
    %DO;
%put * Generating Dates for format "&format"
using SAS format &dtfmt%length(&format). with
Alignment "&align". *;

%DO m = &first %TO &last;
                     %GLOBAL &&format.&m ;

&&format.&m =INTNX("&interval",sysd,&directn.
&m, "&align");

* using &&format.&m to create macro ref.;

CALL SYMPUT("&&format.&m",LOWCASE(TRIM(LEFT
PUT(&&format.&m,&dtfmt%length(&format).)))));
        %END;
    %END;
RUN;
%put>Generating format as &format=&&&format;
                  %GLOBAL displ;
         %LET displ=&&&format;
   %IF %EVAL(&&format.&m > &&format) %THEN
     %DO;
       %DO m = &first %TO &last;
         %LET n = &m;
              %GLOBAL displ&n;
         %LET displ&n = &&format.&m;
 %put > Generating format&m as &&format.&m =
&&&format.&m ;
       %END;
%put>Dates for requested format "&format"
using SAS format &dtfmt%length(&format) are:;
%put>Date &&&format is last updated.;
%put>&n "&interval.s" back generated:
"&&&format.&last" through "&&&format.&first";
%END;
   %ELSE
     %DO;
%put>Dates for &format are: "&&&format" - is
the last updated. There are no extra month
back generated!*;
%END;
%errhandl: %mend anydate;
```

EXAMPLE OF MACRO USE

*Call of macro;

```
   %include "&dmdir/modules/anydate.mod";
        %anydate(format=yyyrmo, interval=m,
        incremnt=-1, last=6);
        %anydate(format=yrmo, interval=m,
        incremnt=-1, last=6);
```

Control message to log:

```
   %put>displ>formats requested:
        <0> format = &displ*;
        <1> format1=&displ1=&&&displ1*;
        <2> format2=&displ2=&&&displ2*;
        <3> format3=&displ3=&&&displ3*;

   %MACRO pull_data(mo_back=0,first=1,last=11);

      %LOCAL m mo_back first last;

   %IF %EVAL(&last >= &first) %THEN
    %DO;
       %DO m = &first %TO &last;
   %put>Creating m=&m* ds=&dsname.&&yrmo&m*
   Data Set from XXX Table EXT_DT=&&yyyrmo&m*;
   %put>  MO_PYMT_AMT&m=MO_PYMT_AMT%EVAL
   (&last+1-&m)*yrmo&m=&&yrmo&m* corresponding
   to %EVAL(&last+1-&m)*;

   PROC SQL;
         CONNECT TO ORACLE (USER=&oralogin
   ORAPW=&orapaswd PATH="@XXX" buffsize=1000);
        CREATE table &dsname.&&yrmo&m
         AS
   SELECT
    PID                       length=6
    informat=11. format=11. LABEL='PERSON ID'
    , CUR_BAL%EVAL(&last+1-&m)  length=6
    informat=9.
   , LS_LT_CHRG%EVAL(&last+1-&m) length=4
   informat=6.
   , LS_CSH_INT%EVAL(&last+1-&m) length=4
   informat=6.
   , MO_CSH_AMT%EVAL(&last+1-&m) length=5
   informat=8.
   , MO_PUR_AMT%EVAL(&last+1-&m) length=5
   informat=8.
   , MO_PMT_AMT%EVAL(&last+1-&m) length=5
   informat=8.
   , CSH%EVAL(&last+1-&m)     length=5
   informat=8. LABEL="CASH AMOUNT"
    , LS_DATE              length=4 informat=6.
    FROM CONNECTION TO ORACLE
    (SELECT
       PID
        , CUR_BAL    CUR_BAL%EVAL(&last+1-&m)
        , LS_LT_CHRG LS_LT_CHRG%EVAL(&last+1-&m)
        , LS_CSH_INT LS_CSH_INT%EVAL(&last+1-&m)
         , MO_CSH_AMT MO_CSH_AMT%EVAL(&last+1-&m)
         , MO_PUR_AMT MO_PUR_AMT%EVAL(&last+1-&m)
        , MO_PMT_AMT  MO_PMT_AMT%EVAL(&last+1-&m)
        , CY_CSH_AMT CSH%EVAL(&last+1-&m)
        , LS_DATE
       FROM XXX.XXXXXXXXXXX
      WHERE &criter and EXT_DT=&&yyyrmo&m);
        QUIT;

        PROC SORT data=&dsname.&&yrmo&m
                                   nodupkey ;
           by pid ;
        RUN;
    %END;
   %put>&dsname.&&yrmo&m last created.
   Generating &m DS: &dsname.&&yrmo&last through
   &dsname.&&yrmo&first*;
   %END;
     %ELSE %DO ;
   %put
   %put>Data Set &dsname.&&yrmo&n is the last
   created. There are no extra Data Sets
   generated: Last less than First! ***;
   %END;
   %mend pull_data;

%pull_data(mo_back = 0, first = 1, last = 6);

  %put yrmo=&yrmo yrmo1=&yrmo1* yrmo2=&yrmo2*
  yrmo3=&yrmo3* yrmo4=&yrmo4* yrmo5=&yrmo5*
  yrmo6=&yrmo6* ;

  * Combine all data sets *;

   DATA curr.&dsname;
     MERGE
        &dsname.&yrmo1(in=last)
        &dsname.&yrmo2
        &dsname.&yrmo3
        &dsname.&yrmo4
        &dsname.&yrmo5
        &dsname.&yrmo6;

        BY pid;

     IF last;

   IF LS_DATE <  &yrmo6 THEN DELETE;

  RUN;
```

EXAMPLE 2

/* creating SAS libname, data sets and variables with embedded in name date stamp from Oracle tables with date in the name. */

```
  %include "&dmdir/modules/anydate.mod";
      %anydate(format=yyyrmo, interval=m,
      incremnt=-1, last=6);

      %anydate(format=yrmo, interval=m,
      incremnt=-1, last=6);

  LIBNAME curr  "&maindir/&yyyrmm";
  . . .
  %IF %EVAL(&last >= &first) %THEN
   %DO;
     %DO m = &first %TO &last;
  %put ** > Currently creating
  *&pgmname.&&yrmo&m* Data Set from
  *XXXX_&&yyyrmo&m* Table **;

  %let n=&m;

   PROC SQL;
    CONNECT TO ORACLE (USER=&oralogin
  ORAPW=&orapaswd PATH="@XXX" buffsize=4096);
    CREATE table curr.&pgmname.&&yrmo&m
    as
  SELECT
    PID length=6 informat=11. label="PID &yrmo"
  , PAYAMT&&yrmo&m length=6  informat=9.2
  label="Payment Amount &&yrmo&m"
  , ENDBAL&&yrmo&m length=6  informat=9.2
  label=" End Bal &&yrmo&m "
  , PDTAG&&yrmo&m  length=3  informat=3.
  label="Profit Driver Tag &&yrmo&m"
    FROM CONNECTION TO ORACLE
   (SELECT
      PID
      , nvl(PAY_AMT,0)   PAYAMT&&yrmo&m
      , nvl(END_BAL,0)   ENDBAL&&yrmo&m
      , PROFIT_SEG       PDTAG&&yrmo&m
   FROM XXX.XXXX_&&yyyrmo&m) ;
   DISCONNECT from ORACLE;
  QUIT;
```

EXAMPLE 3

```
/* does not requires to change dates manually.
Last date to be included must be at least 1-year
back */

   %anydate(format=yyyrmodd, interval=mm,
   incremnt=-1, align=b, last=22);

   %anydate(format=yyyrmodd, interval=mm,
   incremnt=-1, align=b, last=22);

   PROC FORMAT ;

     value $ wavefmt

             low-<"&yyyrmodd22"="Pre-&monyr22"
   "&yyyrmodd22"-<"&yyyrmodd19"="&monyr22.-
   &monyr19"
   "&yyyrmodd19"-<"&yyyrmodd16"="&monyr19.-
   &monyr16"
   "&yyyrmodd16"-<"&yyyrmodd13"="&monyr16.-
   &monyr13"
   "&yyyrmodd13"-<"&yyyrmodd11"="&monyr13.-
   &monyr11"
   "&yyyrmodd11"-high="Post-&monyr11"

     ;

   RUN;
```

CONCLUSION

1. Automatically generating global macro variables representing dates making it unnecessary to manually changing the dates and therefore run routine programs using job schedulers.
2. It allows embedding time stamp into any SAS name, label, automatically creates variables for time series studies.
3. Automatically create customized reports, titles and formats.

REFERENCES

Pete Lund (2001) Make Your Life a Little Easier: A Collection of SAS… Macro Utilities, Proceedings of the twenty-sixths Annual SAS Users Group International Conference, Cary, NC: SAS Institute, Inc., 91-26. (2000) My Favorite Functions, Proceedings of the twenty-fifths Annual SAS Users Group International Conference, Cary, NC: SAS Institute, Inc., 81-25.

Paul D Sherman (2001) Intelligent SAS® Log Manager, Proceedings of the twenty-sixths Annual SAS Users Group International Conference, Cary, NC: SAS Institute, Inc., 108-26.

Frank C. DiIorio (2001) The SAS ® Debugging Primer, Proceedings of the twenty-sixths Annual SAS Users Group International Conference, Cary, NC: SAS Institute, Inc., 54-26.

Arthur L. Carpenter (2000) Functioning With Data Step Functions, Proceedings of the twenty-fifths Annual SAS Users Group International Conference, Cary, NC: SAS Institute, Inc., 57-25.

Armando V. Fabia (2000) Generating Dates Automatically, Proceedings of the twenty-fifths Annual SAS Users Group International Conference, Cary, NC: SAS Institute, Inc., 85-25.

Jay A. Jaffe (1999) SAS Macro Beyond The Basics, Proceedings of the twenty-fourths Annual SAS Users Group International Conference, Cary, NC: SAS Institute, Inc., 149-24.

ACKNOWLEDGEMENTS

The author appreciates the encouragement and support of his wife, Luda. The author also appreciates the support of his employer, Fleet Credit Card Services, Risk Management, his colleagues and is grateful to be able to work on this project.

TRADEMARKS

CONTACT INFORMATION

Anatoly V. Kulinsky
FBF Fleet Credit Card Services
CDRM / Risk Management.
680 Blair Mill Rd.,
Horsham, PA 19044.
Voice (215) 444-3793
Fax (215) 444-3782
E-mail Akulinsky@FleetCC.com