```
%macro mu_setall
(IN_DATA_LIST=
,OUT_DATA=
,TRIGGER_REPLACE_OUT_DATA=N
,SORT_FROM_DATA =
,ABORT_IF_DOES_NOT_EXIST=NO
,HELP =
,DEBUG =
) / store source des='V1.0.0.2';

%*************************************************************************
FILENAME: MU_SETALL.SAS
DEVELOPER: (b) (6)
PLATFORM: PC SAS 9.1.3, 9.2
DATE: 20120207
PURPOSE:  This macro will set a list of data sets together
          to create a new data set.

PARAMETERS:
IN_DATA_LIST    = list of data sets to be set together

OUT_DATA        = name of data set to be created or appended to

TRIGGER_REPLACE_OUT_DATA = whether or not to replace the out_data
                if it already exisits (default = NO)

SORT_FROM_DATA  = pre-sorted data set used to re-sort final data.
                The sort indicator is stored in the data set descriptor
                information and set from a previous sort.  To maintain
                the sort information in a data step, use the sortedby option.
ABORT_IF_DOES_NOT_EXIST= yes/no trigger to abort if a data set listed does not
exist.  (default = NO)

HELP = set to Y(es) to output helpful hints to the log
        (Default = &_default_help)

DEBUG = set to Y(es) to save all intermediate datasets and to turn on mprint. Set
to NO to delete
        all intermediate datasets and to not turn on mprint (maintains original
option setting)
        (Default = &_default_debug)

MODIFIED BY:            DATE:
*************************************************************************;

  %PUT ---------------------------------------------- ;
  %PUT INFO: (&SYSMACRONAME) ;
  %PUT INFO: Version 1.0 ;
  %PUT -START----------------------------------------- ;
```

```
%* MU_SETALL_RC
        0 = ran without error
        1 = missing data set(s)
        2 = aborted due to missing data set(s)
        3 = aborted due to same variable of different types
        4 = aborted due to missing required parameters
        ;


%**** STEP 1 -  Parameter Checks - ensure that required parameters have been
provided;

%let abort = no;
%global &SYSMACRONAME._RC;
%let &SYSMACRONAME._RC = 0;

%let local_abort_if_does_not_exist =
%upcase(%substr(&abort_if_does_not_exist,1,1));

%mu_help_debug

%let local_dataset_list = ;

%***** get current setting of mprint;
%let mprint_setting = %sysfunc(getoption(mprint));
%if &debug = Y %then %do;
        options mprint ;
%end;

%*** if this macro has already been used, then must reset _allcont_ dataset;
        %if %sysfunc(exist(work._allcont_)) ne 0 %then %do;
                %put ALERT_I: DELETING WORK._ALLCONT_;
                proc datasets nolist lib=work;
                        delete _allcont_;
                        run;
                quit;

        %end;



%let &SYSMACRONAME._RC = 4;
%MU_CHECK_REQ_PARAMETERS(parameters_to_check = IN_DATA_LIST OUT_DATA, help=no)
%let &SYSMACRONAME._RC = 0;

%* CHECK FOR LIBREF IN THE OUT_DATA PARAMETER;
        %if &out_data ne %str() %then %do;
             %if %index(&out_data, .) gt 0 %then %do;
               %let out_data_lib = %scan(&out_data, 1, .);
               %let out_data = %scan(&out_data, 2, .);
```

```
            %end;
            %else %do;
              %let out_data_lib = WORK;
            %end;
        %end;


%**** STEP 2 - CHECK THE INPUT DATA SETS AND GET ATTRIBUTES OF ALL THE VARIABLES IN
ALL THE DATA SETS LISTED ;
%* CHECK THE IN_DATA_LIST DATA SETS;
  %macro _check_in_data_list;
        %*   PARSE THE LIST;
        %mu_wordscan(string=&in_data_list, root=inds, numw=numds, delim = %str( ))

        %mu_check_data_and_var_exist(data_to_check = &in_data_list,
abort_if_does_not_exist = no, help=no)


        %* CYCLE THROUGH EACH DATA SET LISTED;
          %global sumrc;
          %let sumrc = 0;
          %do i = 1 %to &numds;

            %* GET THE ATTRIBUTES OF ALL THE VARIABLES IN EACH DATA SET;
              %if %sysfunc(exist(&&inds&i)) ne 0 %then %do;
                %put >>> getting contents of &&inds&i;
                proc contents data=&&inds&i noprint out=_cont_(keep=memname name
type length format);
                run;
                data _cont_;
                  set _cont_;
                  order = &i;
                  name = upcase(name);
                run;

                proc datasets nolist;
                 append base=_allcont_ data=_cont_;
                 run;
                quit;

                %let local_dataset_list = &local_dataset_list _cont_ _allcont_;
              %end;
              %else %do;
                        %put
----------------------------------------------------------------------------
-----------;
                        %put ALERT_C: data set %upcase(&&inds&i) does not exist and
will not be included in the output data set;
                        %if &LOCAL_ABORT_IF_DOES_NOT_EXIST = Y %then %do;
                          %let abort  = YES;
```

```
                          %let &SYSMACRONAME._RC = 2;
                          %put ALERT_R: Macro set to abort if any data set listed
does not exist;
                          %goto local_exit;
                        %end;
                        %else %if &LOCAL_ABORT_IF_DOES_NOT_EXIST = N %then %do;
                          %let &SYSMACRONAME._RC = 1;
                        %end;
                        %put
-------------------------------------------------------------------------------
-----------;
             %end;

             %let sumrc = &sumrc + &&data&i._exist;

          %end;

          %local_exit:
          %let i= &numds;
   %mend _check_in_data_list;
   %_check_in_data_list


   %if &&&SYSMACRONAME._RC = 2 %then %goto exit_and_abort;

   %if &sumrc = 0 %then %do;
        %put ALERT_C: None of the data sets listed in the in_data_list
%upcase(&in_data_list) exist.  ;
        %put ALERT_C: The macro will abort=&LOCAL_ABORT_IF_DOES_NOT_EXIST.;
        %put ALERT_C: Output data set %upcase(&out_data) will not be created or
replaced;
        %let abort = &LOCAL_ABORT_IF_DOES_NOT_EXIST;
        %goto exit_and_abort;
   %end;


%**** STEP 3 - GET THE SORT ORDER;
        %* IF NO SORT_FROM_DATA SPECIFIED, THEN USE THE SORT ORDER FROM THE
        LAST DATA IN THE LIST. ;

        %if &sort_from_data eq %str() %then %do;
          %let sort_from_data=&&inds&numds;
        %end;

        %if %sysfunc(exist(&sort_from_data)) ne 0 %then %do;
          %mu_get_sort_order(&sort_from_data)
          %put sort_order = &sort_order;
        %end;
        %* IF DATA SET SPECIFIED FOR SORT ORDER DOES NOT EXIST, THEN NO SORTING
WILL BE DONE;
```

```
        %else %do;
          %put ALERT_C: Data set %upcase(&sort_from_data) specified in
SORT_FROM_DATA parameter.;
          %put ALERT_C: Data set does not exist and no sort order will be
maintained;
          %LET SORT_ORDER =;
        %end;

%**** STEP 4 - CHECK TO SEE IF THE OUT_DATA ALREADY EXISITS;
        %if %sysfunc(exist(&out_data)) ne 0 %then %do;
          %put
------------------------------------------------------------------------------
-----------;
          %put ALERT_C: OUT_DATA=%upcase(&out_data) already exists;

          %if %upcase(%substr(&TRIGGER_REPLACE_OUT_DATA,1,1)) eq Y %then %do;

             %let out_eq_in  = 0;
             %do i = 1 %to &numds;
               %if %upcase(&out_data) = %upcase(&&inds&i) %then %do;
                 %let out_eq_in = 1;
               %end;
             %end;

             %if &out_eq_in = 0 %then %do;
             %put ALERT_C: %upcase(&out_data) will be deleted and replaced;
              proc datasets nolist lib=&out_data_lib;
                delete &out_data;
                run;
              quit;
             %end;
             %else %if &out_eq_in = 1 %then %do;
              %put ALERT_C: %upcase(&out_data) in %upcase((&in_data_list));
              %put ALERT_C: %upcase(&out_data) will be overwritten;
             %end;

          %end;

          %else %if %upcase(%substr(&TRIGGER_REPLACE_OUT_DATA,1,1)) ne Y %then %do;


             %put ALERT_C: %upcase(&out_data_lib..&out_data) will be added to the
IN_DATA_LIST as the first data set;
               %if %upcase(%substr(&help,1,1)) = Y %then %do;
                %put ALERT_C_HELP: to replace existing data set
%upcase(&out_data_lib..&out_data), set TRIGGER_REPLACE_OUT_DATA to Yes;
               %end;
             %let in_data_list =  &out_data_lib..&out_data &in_data_list;
             %_check_in_data_list
             %if &&&SYSMACRONAME._RC = 2 %then %goto exit_and_abort;
```

```
          %end;

          %put
----------------------------------------------------------------------------------
-----------;
        %end;



%*** STEP 5 - GET A SHORT LIST OF THE VARIABLES BEING SET TOGETHER;
%* ALL THE ATTRIBUTES OF ALL THE VARIABLES BEING SET TOGETHER;
        proc sort data=_allcont_ out=_checktype_ nodupkey;
          by name type;
        run;
%let local_dataset_list = &local_dataset_list _checktype_;




%**** STEP 6 -  IF SAME VARIABLE NAME IS OF DIFFERENT TYPES, MACRO WILL ABORT;
        data _null_;
          set _checktype_;
          by name type;
          if first.name and not last.name then do;
            put
"----------------------------------------------------------------------------------
------------";
            put "ALERT_P: Variable " name " is of different types in input data
sets.";
            put "ALERT_P: &SYSMACRONAME will abort";
            put
"----------------------------------------------------------------------------------
------------";
            call symput('abort', 'yes');
          end;
        run;
        %if &abort = yes %then %do;
          %let &SYSMACRONAME._RC= 3;
          %goto exit_and_abort;
        %end;


%**** STEP 7 - IF SAME VARIABLE NAME HAS DIFFERENT FORMATS, SAS SET WILL USE THE
   FORMAT FROM THE FIRST DATASET LISTED;
        proc sort data=_allcont_ out=_checkformat_ nodupkey;
        by name format order;
        run;
%let local_dataset_list = &local_dataset_list _checkformat_;
```

```
        data check;
                set _checkformat_;
                by name format;
                if first.name and not last.name and "&help" = "Y" then do;
                        put
"------------------------------------------------------------------------------
------------";
                        put "ALERT_I: Variable " name " has different formats in
input data sets.";
                        put "ALERT_I: The first foramt found will be applied";
                        put
"------------------------------------------------------------------------------
------------";
                end;
        run;
%let local_dataset_list = &local_dataset_list check;


%**** STEP 8 -  MACRO WILL RESET LENGTH OF ALL VARIABLES FOR THE LONGEST LENGTH
FOUND
        - NOTE THAT THIS WILL CHANGE THE ORDER OF THE VARIABLES;
        proc sort data=_allcont_ out=_checklength_ nodupkey;
        by name length;
        run;
%let local_dataset_list = &local_dataset_list _checklength_;

        %* CREATE A LENGTH STATEMENT AS A MACRO VARIABLE;
        data _null_;
          set _checklength_ end=eof;
          by name length;
          if last.name;
          length length_statement $ 9999;
          retain length_statement;

          if type = 2 then dollar = '$';

          length_statement =
            trim(left(length_statement)) ||
            " " || trim(left(name)) ||
            " " || dollar ||
            " " || trim(left(put(length, 8.)))
            ;

           if eof then call symput("length_statement",
trim(left(length_statement)));
        run;


%**** STEP 9 -  SET ALL THE DATASETS TOGETHER AND RESET THE LENGTHS OF ALL THE
VARIABLES;
```

```
        data &out_data_lib..&out_data;
          length &length_statement;
          set
           %do i = 1 %to &numds;
           %if %sysfunc(exist(&&inds&i)) %then %do; &&inds&i %end;
           %end;
           ;
        run;


%**** STEP 10 -  IF SORT_FROM_DATA YIELDED A SORT ORDER THEN APPLY IT HERE;
        %if "&sort_order" ne "" %then %do;
          proc sort data=&out_data_lib..&out_data out=&out_data_lib..&out_data;
            by &sort_order;
          run;
        %end;


%*** STEP 11 - END OF PROCESS TASKS****;
%exit_and_abort:


%if %upcase(%substr(&debug, 1, 1)) ne Y %then %do;
        proc datasets lib=work nolist;
                delete &local_dataset_list;
                run;
        quit;
%end;

%if &sumrc ne &numds and &&&SYSMACRONAME._RC ne 3 %then %do;
  %let &SYSMACRONAME._RC = 1 ;
        %if %upcase(%substr(&abort, 1, 1)) = Y %then %do;
          %let &SYSMACRONAME._RC = 2;
        %end;
%end;


options &mprint_setting;

%PUT ---------------------------------------------- ;
%PUT INFO: (&SYSMACRONAME) ;
%PUT INFO: Version 1.0 ;
%PUT INFO: &SYSMACRONAME._RC = &&&SYSMACRONAME._RC;
%PUT -END----------------------------------------- ;


        %if %substr(%upcase(&abort),1,1) = Y %then %do;
            data _null_;
                abort;
            run;
```

```
        %end;
%mend mu_setall;
```