```
%macro mu_align(
     in_data      =
    ,out_data     =
    ,columns      =
    ,colprefix    =
    ,process_if   =
    ,left_spacing = 1000
    ,align        = D
    ,debug        = N
      ) / store source des='V1.0.0.17' ;

%*************************************************************************
**********************
* FILENAME:    MU_ALIGN.SAS
* DEVELOPER:   (b) (6)
* PLATFORM:    SAS 9.1.3, 9.2 on PC
*
* MACROS USED: mu_check_req_parameters, mu_check_data_and_var_exist, mu_checkrc
*              mu_help_debug            md_workinfo
md_clean_and_reset
*
* ASSUMPTIONS: ODS column style attribute PROTECTSPECIALCHARS=OFF is included in
the reporting PROC for
*                the columns provided to this macro at a minimum.
*
*
* DESCRIPTION:
*    Embeds RTF code in variable value for variables entered in COLUMNS parm to
produce decimal aligned values in proc xxxx.
*    Will work with values that do not contain decimals
*    If numeric variables are included, they will be converted to character using
best format and then have RTF code inserted.
*
*
* USAGE NOTES:
* %mu_align(
*       in_data       = mr_pack_pb
*      ,left_spacing  = 1200
*      ,columns       = STAT_VALUE_1_1 STAT_VALUE_1_2 STAT_VALUE_1_3 STAT_VALUE_1_4
STAT_VALUE_1_5 STAT_VALUE_1_6
*          )
*
* %mu_align(
*       in_data       = mr_pack_pb
*      ,colprefix     = STAT_VALUE
*          )
*
*
* PARAMETERS:
* IN_DATA         = input data used containing variables identified in COLUMNS
```

```
(REQUIRED)
* OUT_DATA        = name of output data set, if not entered data set entered in
IN_DATA parm will be used.
* COLUMNS         = name of data set variable(s) to be decimal aligned on report,
list variables separated by a single space
*                   At least one (1) Variable is required to be listed OR COLPREFIX
must have an entry
* COLPREFIX       = If all column names begin with the same set of characters, use
COLPREFIX to select variables
* SELECT_IF       = Allows processing on select rows that meet this condition.
*                   Enter the condition only.  Example:  section = 1
* ALIGN           = D - Decimal, L - Left, C - Center, R - Right.  Default = D
* LEFT_SPACING    = The number of twips to be used to space the decimal aligned
value from the left margin of the column.
*                   The default value is 1000.  The value entered must be a
positive integer.
*********************************************************************
* DATE      INITIALS    MODIFICATION DESCRIPTION
*********************************************************************
*
*********************************************************************
*  © 2011 Pharmaceutical Product Development, Inc.
*  All Rights Reserved.
*********************************************************************************
**********************************;

   %local mname pretext varstring varnames _vars rnamevar columnsx loop badls
ls_value ls_vals   lsarray add_ls
          cloop group1  group2 _d missing  value i        varlen   help twips_len
drop_ls  var_labels mvars
          lib   ds      libout    keepds _checkrc;

   %let mname = &sysmacroname;


   %mu_help_debug ;

   %* -- Get current setting of mprint and if set to DEBUG mode turn mprint on,
Create list of current work data sets;
   %md_workinfo(debug = &debug );

   %if &debug = Y %then %mu_checkrc(condition = 1=1
              ,msg1      = Begin processing for Macro &mname
              ,rcprefix  = &mname
              ,rc        = 0
                );


   %* -- Check that in_data is entered and exists --;
   %mu_check_req_parameters(parameters_to_check = IN_DATA );
```

```
   %* -- COLUMNS or COLPREFIX must have a value entered --;
   %mu_checkrc(condition = %bquote(&columns) = %str( ) and %bquote(&colprefix) =
%str( )
                      ,setabort  = Y
                      ,rcprefix  = &mname
                      ,rc        = 7
                      ,msg1      = %str(A value MUST be entered in either the
COLUMNS or COLPREFIX parameter)
                      ,msg2      = %str(COLUMNS ===> &columns , COLPREFIX =
&colprefix)
                        );

   %* Determine if work data set or permanent data set was entered in IN_DATA for
SQL processing --;
   %let in_data = %upcase(&in_data);
   %if %index(%bquote(&in_data), %str(.)) >0 %then %do;
       %let lib = %scan(&in_data, 1, %str(.));
       %let ds  = %scan(&in_data, 2, %str(.));
   %end;
   %else %do;
       %let lib = WORK;
       %let ds  = &in_data;
   %end;

   %let lib = %upcase(&lib);
   %let ds  = %upcase(&ds);


   %* -- Get sort order of input data set --;
   %mu_get_sort_order(&in_data);

   %* -- Get column names based upon COLPREFIX entry --;
   %if %bquote(&colprefix) ne %str( ) %then %do;

        %let colprefix =
%sysfunc(compress(%sysfunc(tranwrd(%bquote(&colprefix),%str(_),%str(/_))), %str(,)
));

       %* -- Multiple values may have been entered for COLPREFIX --;
       %mu_wordscan(string= &colprefix, numw=nprefix, root=cprefix);

       proc sql noprint;
             select  upcase(name)
               into :columnsx separated by ' '
               from  dictionary.columns
              where  libname = "&lib"
                and  memname = "&ds"
                and  (
```

```
            %do loop = 1 %to &nprefix;                 %* Loop thru multiple
prefixes entered;
                %if &loop > 1 %then %str( or );
                     upcase(name)   like "%upcase(&&cprefix&loop.%)" escape "/"
            %end;
                      );

       quit;

       %* -- If no variables were found containing the COLPREFIX entered, ALERT and
ABORT --;
       %mu_checkrc(condition = %bquote(&columnsx) = %str( )
                   ,setabort  = Y
                   ,rcprefix  = &mname
                   ,msg1      = %str(No variables were found with the prefix
[&colprefix])
                   ,msg2      = %str(Check value entered in parameter COLPREFIX)
                     );

   %end;

   %* -- Confirm values entered in COLUMNS parm exist in IN_DATA, otherwise program
will stop --;
   %if %bquote(&columns) ne %str( ) %then %do;
       %let columns = %upcase(&columns);
       %let columns = %sysfunc(compress(%bquote(&columns), %str(,) ));

       %* -- this macro will run mu_wordscan and create macro array of columns
variables and numvar (count) of vars;
       %mu_check_data_and_var_exist(
           data_to_check             = &in_data
          ,vars_to_check_in_all_data = &columns
          ,abort_if_does_not_exist   = YES
            );
   %end;
   %else %let numvar = 0;


   %* -- If no output data set was entered in OUT_DATA, use data set specified in
IN_DATA parm --;
   %if %md_notnull(out_data) = 0  %then %let out_data = &in_data;    %* --
04APR2016 (b) (6) updated to use md_notnull -;

   %* -- set RTF code for pretext -------------------------;
   %let align = %substr(%lowcase(&align),1,1);

   %mu_checkrc(condition = %sysfunc(verify(%bquote(&align), %str(lcrd))) >0
              ,rcprefix  = &mname
              ,rc        = 1
              ,msg1      = %str(A missing or incorrect value was entered in ALIGN
```

```
===> &align)
              ,msg2      = %str(ALIGN will be set to Decimal Alignment (D))
                );

        %if &&&mname._RC=1 %then %let _checkrc=1;

   %if &align = d or &&&mname._RC = 1 %then %do;

       %* -- Left spacing
------------------------------------------------------------;

       %* -- Null value entered, set default --;
       %if %bquote(&left_spacing) = %str( ) %then %let left_spacing = 1000;

       %else %do;

             %* -- if more than one variable was entered in COLUMNS parm create
macro array of values in left_spacing;
             %if &numvar > 1 %then %do;

                 %* -- Pull LSx macro variables that already exist as GLOBAl macro
vars;
                 proc sql noprint;
                      select name
                        into :mvars separated by ' '
                        from dictionary.macros
                       where scope     = 'GLOBAL'
                         and name   like 'LS%'
                        ;
                 quit;

                 %if %bquote(&mvars) ne %str( ) %then %do;
                     %* -- Delete LSx macro variables --;
                     %symdel &mvars;
                 %end;

                 %mu_wordscan(string = %bquote(&left_spacing), root=ls, numw =
lsnum);
             %end;

             %else %do;
                 %let ls1 = &left_spacing;
                 %let lsnum = 1;
             %end;

             %* -- Confirm Left spacing needs is a postive integer, otherwise ALERT
and stop --;
             %do lsck = 1 %to &lsnum;
                 %if %sysfunc(verify(%bquote(&&ls&lsck), %str(1234567890) )) > 0
%then %let badls = &badls &&ls&lsck ;
```

```
            %end;

            %mu_checkrc(condition = %length(&badls) > 0
                       ,setabort  = Y
                       ,rcprefix  = &mname
                       ,rc        = 8
                       ,msg1      = %str(LEFT_SPACING needs to be a positve
integer value, value entered ===> &left_spacing)
                       ,msg2      = %str(Invalid values are:  &badls)
                         );
       %end;

       %if &lsnum = 1 %then %let pretext = %str(\ql\tqdec\tx&left_spacing);   %*
only one value for left_spacing was entered;
       %else                %let pretext = %str(\ql\tqdec\tx);                %*
mult values entered, will be appended in data step;

       %let twips_len = 8;
   %end;

   %else %do;
       %let pretext   = %str(\q&align);                          %* set left,
right, or center justification only;
       %let twips_len = 0;
       %let lsnum     = 0;
   %end;

   %* -- Combine variable names from both COLUMNS parm and COLPREFIX parm --;
   %let columns = %upcase(%cmpres(&columns  &columnsx));

   %* -- Use MU_QUOTER to place quotes about each variable name --;
   %let col_sql = %mu_quoter(%trim(&columns), dlm = %str(,) );
   %let col_sql = %sysfunc(compbl(%bquote(&col_sql) ));

   data &out_data;
    set &in_data;
   run;


   %let group1 = CHAR;
   %let group2 = NUM;

   %* -- Process data set once for character variables and once for numeric
variables --;
   %do i = 1 %to 2;
       %let lsarray =;
       %let ls_vals =;
       %let add_ls  =;
       %let drop_ls =;
```

```
      %* --  Pull variables by type (character or numeric) ---;
      proc sql noprint;

          select  upcase(name),               compress('_'||name),
compbl(name||' = _'||name), max(length)
            into  :varnames separated by ' ', :_vars separated by ' ' , :rnamevar
separated by ' ' , :mlength

            from  dictionary.columns

           where  libname    = "&lib"
             and  memname    = "&ds"
             and  upcase(trim(name)) in(&col_sql )
             and  upcase(type)       = "&&group&i"
                ;

      %* -- If any variables of type being processed are found, then add rtf code
--;
      %if %bquote(&varnames) ne %str( ) %then %do;

          %* -- ALIGN = D: More than one LEFT_SPACING value was entered in parm,
match it to NUMVAR --;
          %if &lsnum > 1 %then %do;

              %* -- Place all variables of type being processed into macro
variable array --;
              %mu_wordscan(string = %bquote(&varnames), root=vn, numw=nvn);

              %do _vars_ = 1 %to &nvn;        %* -- All variables of this type --;

                  %let ls_value = &ls1;       %* -- Initialize spacing to first
value entered --;

                  %do loop = 1 %to &numvar;   %* -- All variables entered in
COLUMNS parm --;

                      %if &&var&loop = &&vn&_vars_ %then %do;  %* type variable is
variable entered in COLUMNS parm;

                          %if %symexist(ls&loop) = 0 %then %do;
                              %let ls_value = &ls1;       %* -- No corresponding
left_spacing value;
                              %mu_checkrc(condition = 1=1
                                         ,rcprefix  = &mname
                                         ,rc        = 2
                                         ,msg1      = %str(&mname:  No value for
LEFT_SPACING was provided for this variable  ===> &&var&loop)
                                         ,msg2      = %str(&mname:  LEFT_SPACING
be set to first LEFT_SPACING value entered   ===> &ls1)
                                          );
```

```
                        %end;
                        %else                           %let ls_value =
&&ls&loop;  %* -- Match variable with spacing entered;
                     %end;
                  %end;

                  %let ls_vals = &ls_vals &ls_value;   %* -- Put each spacing
value in ordered string ;
              %end;

              %* -- Build left spacing array -------------------;
              %if %bquote(&ls_vals) ne %str( ) %then %do;
                  %let lsarray   = %str( array xlefts  ls1 - ls&nvn (&ls_vals););
                  %let add_ls    = %str( ||strip(put(xlefts, 8.)) );
                  %let drop_ls   = %str( ls1 - ls&nvn );
              %end;
           %end;

           %if %str(&&group&i) = CHAR %then %do;
              %let _d      = $;
              %let missing = %str(' ');
              %let value   = %str( strip(xcolsxn) );
           %end;
           %else %do;
              %let _d      = ;
              %let value   = %str( strip(put(xcolsxn, best.)) );
              %let missing = %str(.);
           %end;

           %let varlen =  %eval(&mlength + %length(&pretext) + &twips_len) ;  %* --
Determine new variable length  --;
           %if %bquote(&process_if) = %str( ) %then %let process_if =
%str(_dummy_var_ = 1);

           data &out_data(drop   = &_vars _dummy_var_  &drop_ls);
            set &out_data(rename = (&rnamevar ));
           retain _dummy_var_ 1;

               array xcolsxn &_d       &_vars;
               array xcolsx  $&varlen  &varnames;
               &lsarray

               do over xcolsx;

                  IF &process_if THEN DO;
                     if      xcolsxn = &missing            then xcolsx = ' ';
                    %if %str(&_d) = %str($) %then %str(
                     else if substr(xcolsxn, 1, 2 ) = '\q' then xcolsx = &value;
);
                     else                                       xcolsx = "&pretext"
```

```
&add_ls  || ' ' || &value ;
                  END;
                  ELSE                                              xcolsx = &value ;

              end;
          run;
       %end; %*  %bquote(&&group&i) ne %str( ) criteria;
   %end;  %* loop for char/num;


   %* -- Put labels (back)  on variables  ---;
   proc sql noprint;
        select  compbl(name || " = '" || trim(label) ||"'" )
          into :var_labels separated by ' '

          from  dictionary.columns

         where  libname = "&lib"
           and  memname = "&ds"
           and  upcase(name) in(&col_sql )
           and  label   is not null;
   quit;


   %if %index(%bquote(&out_data), %str(.)) >0 %then %let libout = %scan(&out_data,
1, %str(.));  %* -- 01Apr2016 (b) (6) moved outside of loop below --;
   %else                                              %let libout = WORK;


   %* -- Apply SORTEDBY to output data set and variable labels --;

   %if &MU_GET_SORT_ORDER_RC =0 or  %bquote(&var_labels) ne %str( ) %then %do;

       proc datasets lib=&libout nolist;
            modify &out_data;
            %if %bquote(&var_labels) ne %str( ) %then %str( label &var_labels;);
            run;
       quit;
       proc sort data = &out_data;
         by &sort_order;
       run;

   %end;

   %if %bquote(&debug) = Y %then %do;
       title10 "&MNAME: debug print";
       proc print data=&out_data(obs=30) label;
         var &columns;
       run;
       title10;
```

```
   %end;

   %if &_checkrc=1 and &&&mname._RC=0 %then %let &&mname._RC=1;

   %* -- Based upon settings, remove work data sets not included below --;
   %if %upcase(&libout) = WORK %then %let keepds = &out_data;    %* -- 01Apr2016
(b) (6), created KEEPDS and removed 2nd call to md_clean_and_reset --;

   %md_clean_and_reset(
       debug     = &debug
      ,_workdata = %str(&WORK_DATASETS_DATA &keepds )
      ,resetmprint = &mprint_setting
        )


    %if &debug = Y %then %mu_checkrc(condition = %str(&debug) = Y
               ,msg1      = %str(End processing for Macro &mname)
                 );
%mend mu_align;
```