HOW TO SET A DOXY_ADJUSTED_ERROR IN CORIOLIS NETCDF FILES

This note, that follows #RD4 recommendations, explains how the DOXY_ADJUSTED_ERROR information should be sent to Coriolis (to Vincent.Bernard@ifremer.fr) so that it will be propagated in Real Time into the profiles files and how the SCIENTIFIC_CALIB_COMMENT (of the DOXY parameter) will be set accordingly.

The PI must provide the estimated DOXY_ADJUSTED_ERROR of their floats together with the estimation method used (designated with a specific number from 0 to 3, see below).

The method number is then used by the decoder: 1- to determine the method used to propagate the provided error at the profile levels; 2- to set the SCIENTIFIC_CALIB_COMMENT associated to DOXY parameter.

#RD4 : DOI: http://dx.doi.org/10.13155/46542

History

Date (dd/mmm/yyyy)	Comment
02/04/2020	Creation of the document by V. Racapé and Catherine Schmechtig
19/04/2021	Update to include the time dependence in the propagation error method as recommended by #RD4
21/01/2022	Add information regarding the sub method # 3_2 : Adjustment based on last valid DM adjustment that includes a refined pressure correction coefficient – Matlab code has been updated accordingly

1. LIST OF AVAILABLE PROPAGATION ERROR METHODS

Case 1 : Constant error in PPOX

Propagation ERROR Method # = 1

Description of the method: propagation error of 10mbar by default or provided by PI in mbar

DOXY_ADJUSTED_ERROR = [X] µmol/kg is recomputed from CALIB_RT_ADJUSTED_ERROR

Case 2 : increasing error in time in PPOX

Propagation ERROR Method # = 2

Description of the method: propagation error of 10mbar by default or provided by PI in mbar with an increase of 1mbar per year since the last (RT or DM) adjustment

DOXY_ADJUSTED_ERROR = [X] µmol/kg is recomputed from CALIB_RT_ADJUSTED_ERROR

2. LIST OF AVAILABLE ESTIMATION METHODS

Method # = 1

Description of the method: gain estimated from the comparison between in water PSAT or PPOX from float and PSAT or PPOX from WOA at most in the upper 20 dbar of the water column. WOA PPOX is computed from WOA PSAT and from TEMP and PSAL float data at the atmospheric pressure of 1 atm.

Case 1_1: Adjustment by comparison of in water float data to WOA based on PSAT or PPOX, error in PPOX

SCIENTIFIC_CALIB_COMMENT = "DOXY_ADJUSTED is computed from an adjustment of in water PSAT or PPOX float data at surface by comparison to WOA PSAT climatology or WOA PPOX in using $PSAT_{WOA}$ and TEMP and $PSAL_{float}$ at 1 atm, DOXY_ADJUSTED_ERROR is computed from a PPOX_ERROR of xx.x mbar"

Propagation ERROR Method # = 1

Case 1_2: Adjustment by comparison of in water float data to WOA based on PSAT or PPOX, error in PPOX increasing with time

SCIENTIFIC_CALIB_COMMENT = "DOXY_ADJUSTED is computed from an adjustment of in water PSAT or PPOX float data at surface by comparison to woaPSAT climatology or woaPPOX{woaPSAT,floatTEMP,floatPSAL} at 1 atm, DOXY_ADJUSTED_ERROR is computed from a PPOX_ERROR of xx.x mbar +1mb/year"

Propagation ERROR Method # = 2

Method # = 2

Description of the method: gain estimated from the comparison between in air $PPOX_{float}$ and $PPOX_{NCEP}$.

Case 2_1: Adjustment by comparison of in air float data to NCEP reanalysis atmospheric data based on PPOX, error in PPOX

SCIENTIFIC_CALIB_COMMENT = "DOXY_ADJUSTED is estimated from an adjustment of in air PPOX float data by comparison to NCEP reanalysis, DOXY_ADJUSTED_ERROR is recomputed from a PPOX_ERROR = xx.x mbar"

Propagation ERROR Method # = 1

Case 2_2: Adjustment by comparison of in air float data to NCEP reanalysis atmospheric data based on PPOX, error in PPOX increasing with time

SCIENTIFIC_CALIB_COMMENT = "DOXY_ADJUSTED is estimated from an adjustment of in air PPOX float data by comparison to NCEP reanalysis, DOXY_ADJUSTED_ERROR is recomputed from a PPOX_ERROR = xx.x mbar with an increase of 1mbar/year"

Propagation ERROR Method # = 2

Method # = 3

Description of the method: Adjustment information (SLOPE – OFFSET – DRIFT – INCLINE_T) provided by the last valid cycle with DM adjustment. In some cases, additional correction beyond the common scientific equation could be recorded in the SCIENTIFIC_CALIB section by the expansion of the N_CALIB dimension. In this way, the sub method 3_2 takes into consideration the fine-tuning of the pressure correction coefficient as well.

Case 3_2: Adjustment based on last valid DM adjustment, error in PPOX increasing with time

SCIENTIFIC_CALIB_COMMENT = "DOXY_ADJUSTED is estimated from the last valid cycle with DM adjustment, DOXY_ADJUSTED_ERROR is recomputed from a PPOX_ERROR = xx.x mbar with an increase of 1mbar/year"

Propagation ERROR Method # = 2

Case 3_2_2 Adjustment based on last valid DM adjustment that includes a refined pressure correction coefficient, error in PPOX increasing with time

SCIENTIFIC_CALIB_COMMENT = "DOXY_ADJUSTED is estimated from the last valid cycle with DM adjustment that includes a refined pressure correction coefficient, DOXY_ADJUSTED_ERROR is recomputed from a PPOX_ERROR = xx.x mbar with an increase of 1mbar/year"

Propagation ERROR Method # = 2

3. MATLAB CODE OF DOXY REAL TIME ADJUSTMENT

```
% ------------------------------------------------------------------------------
% Perform real time adjustment on DOXY profile data.
%
% SYNTAX :
%  [o_tabProfiles, o_tabTrajNMeas, o_tabTrajNCycle] = ...
%    compute_rt_adjusted_doxy(a_tabProfiles, a_tabTrajNMeas, a_tabTrajNCycle, a_launchDate,
a_decoderId)
%
% INPUT PARAMETERS :
%   a_tabProfiles   : input profile structures
%   a_tabTrajNMeas  : input N_MEASUREMENT trajectory data
%   a_tabTrajNCycle : input N_CYCLE trajectory data
%   a_launchDate    : float launch date
%   a_decoderId     : float decoder Id
%
% OUTPUT PARAMETERS :
%   o_tabProfiles   : output profile structures
%   o_tabTrajNMeas  : output N_MEASUREMENT trajectory data
%   o_tabTrajNCycle : output N_CYCLE trajectory data
%
% EXAMPLES :
%
% SEE ALSO :
% AUTHORS  : Jean-Philippe Rannou (Altran)(jean-philippe.rannou@altran.com)
% ------------------------------------------------------------------------------
% RELEASES :
%   07/03/2019 - RNU - creation
% ------------------------------------------------------------------------------
function [o_tabProfiles, o_tabTrajNMeas, o_tabTrajNCycle] = ...
   compute_rt_adjusted_doxy(a_tabProfiles, a_tabTrajNMeas, a_tabTrajNCycle, a_launchDate,
a_decoderId)

% output parameters initialization
o_tabProfiles = a_tabProfiles;
o_tabTrajNMeas = a_tabTrajNMeas;
o_tabTrajNCycle = a_tabTrajNCycle;

% current float WMO number
global g_decArgo_floatNum;

% arrays to store RT offset information
global g_decArgo_rtOffsetInfo;

% global default values
global g_decArgo_dateDef;
global g_decArgo_nbHourForProfDateCompInRtOffsetAdj;
global g_decArgo_janFirst1950InMatlab;

% to store information on DOXY adjustment
global g_decArgo_paramProfAdjInfo;
global g_decArgo_paramProfAdjId;

% TRAJ 3.2 file generation flag
global g_decArgo_generateNcTraj32;


% look for DOXY profiles
noDoxyProfile = 1;
for idProf = 1:length(o_tabProfiles)
   if (any(strcmp({o_tabProfiles(idProf).paramList.name}, 'DOXY')))
      noDoxyProfile = 0;
      break
   end
```

```
end
if (noDoxyProfile)
   % under the assumption that no DOXY profiles means that there is no DOXY
   % measurement in the trajectory
   return
end

% retrieve information on DOXY adjustement in RT_OFFSET information of META.json file
doSlope = '';
doOffset = '';
doDrift = '';
doInclineT = '';
doCorPres = '';
doDate = '';
doAdjError = '';
doAdjErrorStr = '';
doAdjErrMethod = '';
if (~isempty(g_decArgo_rtOffsetInfo))
   for idF = 1:length(g_decArgo_rtOffsetInfo.param)
      if (strcmp(g_decArgo_rtOffsetInfo.param{idF}, 'DOXY'))
         % mandatory fields
         doSlope = g_decArgo_rtOffsetInfo.slope{idF};
         doOffset = g_decArgo_rtOffsetInfo.value{idF};
         doDate = g_decArgo_rtOffsetInfo.date{idF};
         % not mandatory fields
         if (isfield(g_decArgo_rtOffsetInfo, 'adjError'))
            doAdjError = g_decArgo_rtOffsetInfo.adjError{idF};
            doAdjErrorStr = g_decArgo_rtOffsetInfo.adjErrorStr{idF}{:};
            doAdjErrMethod = g_decArgo_rtOffsetInfo.adjErrorMethod{idF}{:};
         end
         % new fields (possibly not filled for old adjustments)
         doDrift = 0;
         if (isfield(g_decArgo_rtOffsetInfo, 'drift'))
            doDrift = g_decArgo_rtOffsetInfo.drift{idF};
         end
         doInclineT = 0;
         if (isfield(g_decArgo_rtOffsetInfo, 'inclineT'))
            doInclineT = g_decArgo_rtOffsetInfo.inclineT{idF};
         end
         doCorPres = nan;
         if (isfield(g_decArgo_rtOffsetInfo, 'doCorPres'))
            doCorPres = g_decArgo_rtOffsetInfo.doCorPres{idF};
         end
         break
      end
   end
end

if (~isempty(doSlope))

   profDateList = [o_tabProfiles.date];
   profDateList(profDateList == g_decArgo_dateDef) = [];

   if (any((profDateList + g_decArgo_nbHourForProfDateCompInRtOffsetAdj/24) >= doDate))

      % some cases need the PPOX_ERROR to increase with time
      startDateToIncreasePpoxErrorWithTime = '';
      if (~isnan(doAdjError))
         switch (doAdjErrMethod)
            case {'1_1', '2_1'}
               startDateToIncreasePpoxErrorWithTime = '';
            case {'1_2', '2_2', '3_2', '3_2_2'}
               startDateToIncreasePpoxErrorWithTime = doDate;
         end
      end

      % basic adjustment information for NetCDF files
```

```
      % default equation (same as for case '1_1', '1_2', '2_1', '2_2' or '3_2')
      equation = ['PPOX=f(DOXY), ' ...
         'PPOX_DOXY_ADJUSTED=(SLOPE*(1+DRIFT/100*(profile_date_juld-
launch_date_juld)/365)+INCLINE_T*TEMP)*(PPOX_DOXY+OFFSET), ' ...
         'DOXY_ADJUSTED=f(PPOX_DOXY_ADJUSTED)'];
      coefficient = sprintf('OFFSET = %.2f, SLOPE = %.4f, DRIFT = %.3f, INCLINE_T = %.6f,
launch_date_juld = %s', ...
         doOffset, doSlope, doDrift, doInclineT, datestr(a_launchDate +
g_decArgo_janFirst1950InMatlab, 'yyyymmddHHMMSS'));
      if (~isnan(doAdjError))
         switch (doAdjErrMethod)
            case {'1_1', '1_2', '2_1', '2_2', '3_2'}
               equation = ['PPOX_DOXY=f(DOXY), ' ...
                  'PPOX_DOXY_ADJUSTED=(SLOPE*(1+DRIFT/100*(profile_date_juld-
launch_date_juld)/365)+INCLINE_T*TEMP)*(PPOX_DOXY+OFFSET), ' ...
                  'DOXY_ADJUSTED=f(PPOX_DOXY_ADJUSTED)'];
               coefficient = sprintf('OFFSET = %.2f, SLOPE = %.4f, DRIFT = %.3f, INCLINE_T =
%.6f, launch_date_juld = %s', ...
                  doOffset, doSlope, doDrift, doInclineT, datestr(a_launchDate +
g_decArgo_janFirst1950InMatlab, 'yyyymmddHHMMSS'));
            case {'3_2_2'}
               equation = ['DOXY_COR_PRES=DOXY*(1+DO_COR_PRES*PRES/1000), '...
                  'PPOX_DOXY=f(DOXY_COR_PRES), ' ...
                  'PPOX_DOXY_ADJUSTED=(SLOPE*(1+DRIFT/100*(profile_date_juld-
launch_date_juld)/365)+INCLINE_T*TEMP)*(PPOX_DOXY+OFFSET), ' ...
                  'DOXY_ADJUSTED=f(PPOX_DOXY_ADJUSTED)'];
               coefficient = sprintf('OFFSET = %.2f, SLOPE = %.4f, DRIFT = %.3f, INCLINE_T =
%.6f, DO_COR_PRES = %.4f, launch_date_juld = %s', ...
                  doOffset, doSlope, doDrift, doInclineT, doCorPres, datestr(a_launchDate +
g_decArgo_janFirst1950InMatlab, 'yyyymmddHHMMSS'));
         end
      end
      comment = '';
      if (~isnan(doAdjError))
         switch (doAdjErrMethod)
            case '1_1'
               comment = sprintf(['DOXY_ADJUSTED is computed from an adjustment ' ...
                  'of in water PSAT or PPOX float data at surface by comparison to woaPSAT '
...
                  'climatology or WOA PPOX in using woaPSAT and floatTEMP and PSAL at 1 atm, '
...
                  'DOXY_ADJUSTED_ERROR is computed from a PPOX_ERROR of %s mbar'],
doAdjErrorStr);
            case '1_2'
               comment = sprintf(['DOXY_ADJUSTED is computed from an adjustment ' ...
                  'of in water PSAT or PPOX float data at surface by comparison to woaPSAT '
...
                  'climatology or woaPPOX{woaPSAT,floatTEMP,floatPSAL} at 1 atm, ' ...
                  'DOXY_ADJUSTED_ERROR is computed from a PPOX_ERROR of %s mbar +1mb/year'],
doAdjErrorStr);
            case '2_1'
               comment = sprintf(['DOXY_ADJUSTED is estimated from an adjustment ' ...
                  'of in air PPOX float data by comparison to NCEP reanalysis, ' ...
                  'DOXY_ADJUSTED_ERROR is recomputed from a PPOX_ERROR = %s mbar'],
doAdjErrorStr);
            case '2_2'
               comment = sprintf(['DOXY_ADJUSTED is estimated from an adjustment ' ...
                  'of in air PPOX float data by comparison to NCEP reanalysis, ' ...
                  'DOXY_ADJUSTED_ERROR is recomputed from a PPOX_ERROR = %s mbar ' ...
                  'with an increase of 1mbar/year'], doAdjErrorStr);
            case '3_2'
               comment = sprintf(['DOXY_ADJUSTED is estimated from the last valid cycle ' ...
                  'with DM adjustment, DOXY_ADJUSTED_ERROR is recomputed from a ' ...
                  'PPOX_ERROR = %s mbar with an increase of 1mbar/year'], doAdjErrorStr);
            case '3_2_2'
               comment = sprintf(['DOXY_ADJUSTED is estimated from the last valid cycle ' ...
```

```
                  'with DM adjustment that includes a refined pressure correction coefficient,
', ...
                  'DOXY_ADJUSTED_ERROR is recomputed from a ' ...
                  'PPOX_ERROR = %s mbar with an increase of 1mbar/year'], doAdjErrorStr);
            otherwise
               fprintf('ERROR: Float #%d: input CALIB_RT_ADJ_ERROR_METHOD (''%s'') of DOXY
adjustment is not implemented yet - SCIENTIFIC_CALIB_COMMENT of DOXY parameter not set\n', ...
                  g_decArgo_floatNum, ...
                  doAdjErrMethod);
         end
      end
      date = datestr(doDate+g_decArgo_janFirst1950InMatlab, 'yyyymmddHHMMSS');

      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      % adjust DOXY profile data
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      for idProf = 1:length(o_tabProfiles)
         profile = o_tabProfiles(idProf);
         if (any(strcmp({profile.paramList.name}, 'DOXY')) && ...
               (profile.date ~= g_decArgo_dateDef) && ...
               ((profile.date + g_decArgo_nbHourForProfDateCompInRtOffsetAdj/24) >= doDate))

            % retrieve associated profiles (needed for 'real' BGC floats since
            % PTS are in separate profiles)
            idProfs = find(([o_tabProfiles.outputCycleNumber] == profile.outputCycleNumber) &
...
               ([o_tabProfiles.direction] == profile.direction) & ...
               ([o_tabProfiles.sensorNumber] < 100)); % AUX profiles should not be considered

            % adjust DOXY for this profile
            [ok, profile] = adjust_doxy_profile( ...
               profile, o_tabProfiles(setdiff(idProfs, idProf)), ...
               doCorPres, doSlope, doOffset, doDrift, doInclineT, ...
               doAdjError, startDateToIncreasePpoxErrorWithTime, a_launchDate, a_decoderId);
            if (ok)
               profile.rtParamAdjIdList = [profile.rtParamAdjIdList g_decArgo_paramProfAdjId];
               o_tabProfiles(idProf) = profile;

               % store profile adjustment information for NetCDF file
               if (profile.direction == 'A')
                  direction = 2;
               else
                  direction = 1;
               end

               g_decArgo_paramProfAdjInfo = [g_decArgo_paramProfAdjInfo;
                  g_decArgo_paramProfAdjId profile.outputCycleNumber direction ...
                  {'DOXY'} {equation} {coefficient} {comment} {date}];
               g_decArgo_paramProfAdjId = g_decArgo_paramProfAdjId + 1;
            end
         end
      end

      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      % adjust DOXY trajectory data
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      if (g_decArgo_generateNcTraj32 ~= 0)

         idProfToAdjust = find(([o_tabProfiles.date] ~= g_decArgo_dateDef) & ...
            (([o_tabProfiles.date] + g_decArgo_nbHourForProfDateCompInRtOffsetAdj/24) >=
doDate));
         firstCyToAdjust = min([o_tabProfiles(idProfToAdjust).outputCycleNumber]);

         idTrajToAdjust = find([o_tabTrajNMeas.outputCycleNumber] >= firstCyToAdjust);
         adjFlag = 0;
```

```
         for idTraj = idTrajToAdjust
            for idMeas = 1:length(o_tabTrajNMeas(idTraj).tabMeas)
               tabMeas = o_tabTrajNMeas(idTraj).tabMeas(idMeas);
               if (~isempty(tabMeas.paramList) && ...
                     any(strcmp({tabMeas.paramList.name}, 'DOXY')))

                  % adjust DOXY for this measurement
                  [ok, tabMeas] = adjust_doxy_traj_meas(tabMeas, ...
                     doCorPres, doSlope, doOffset, doDrift, doInclineT, ...
                     doAdjError, startDateToIncreasePpoxErrorWithTime, a_launchDate, ...
                     o_tabTrajNMeas(idTraj), idMeas, a_decoderId);
                  if (ok)
                     o_tabTrajNMeas(idTraj).tabMeas(idMeas) = tabMeas;
                     adjFlag = 1;

                     % store trajectory adjustment information for NetCDF file
                     store_traj_adj_info(3, o_tabTrajNMeas(idTraj).outputCycleNumber, ...
                        'DOXY', equation, coefficient, comment, date);
                  end
               end
            end
         end

         % update DATA_MODE
         if (adjFlag)
            if (any([o_tabTrajNCycle.dataMode] ~= 'A'))
               idCyList = find([o_tabTrajNCycle.dataMode] ~= 'A');
               for idCy = 1:length(idCyList)
                  idStruct = find([o_tabTrajNMeas.outputCycleNumber] ==
o_tabTrajNCycle(idCyList(idCy)).outputCycleNumber); % nominal case: only one
                  for idS = 1:length(idStruct)
                     tabTrajNMeas = o_tabTrajNMeas(idStruct(idS));
                     if (any([tabTrajNMeas.tabMeas.paramDataMode] == 'A'))
                        o_tabTrajNCycle(idCyList(idCy)).dataMode = 'A';
                        break
                     end
                  end
               end
            end
         end
      end
   end
end

return

% ------------------------------------------------------------------------------
% Perform real time adjustment on one DOXY measurement data.
%
% SYNTAX :
%  [o_ok, o_tabMeas] = adjust_doxy_traj_meas(a_tabMeas, ...
%    a_doCorPres, a_slope, a_offset, a_doDrift, a_doInclineT, ...
%    a_adjError, a_adjDate, a_launchDate, a_trajNMeas, a_idMeas, a_decoderId)
%
% INPUT PARAMETERS :
%   a_tabMeas    : input DOXY measurement structure
%   a_doCorPres  : coefficient for DOXY correction f(PRES) - in CASE 3_2_2 only
%   a_slope      : slope to be used for PPOX_DOXY adjustment
%   a_offset     : offset to be used for PPOX_DOXY adjustment
%   a_doDrift    : drift to be used for PPOX_DOXY adjustment
%   a_doInclineT : incline_t to be used for PPOX_DOXY adjustment
%   a_adjError   : error on PPOX_DOXY adjusted values
%   a_adjDate    : start date to apply adjustment
%   a_launchDate : float launch date
%   a_trajNMeas  : associated cycle measurements
%   a_idMeas     : index of current measurement in cycle measurements
%   a_decoderId  : float decoder Id
```

```
%
% OUTPUT PARAMETERS :
%   o_ok      : 1 if the adjustment has been performed, 0 otherwise
%   o_tabMeas : output DOXY measurement structure
%
% EXAMPLES :
%
% SEE ALSO :
% AUTHORS  : Jean-Philippe Rannou (Altran)(jean-philippe.rannou@altran.com)
% ------------------------------------------------------------------------------
% RELEASES :
%   09/01/2021 - RNU - creation
% ------------------------------------------------------------------------------
function [o_ok, o_tabMeas] = adjust_doxy_traj_meas(a_tabMeas, ...
   a_doCorPres, a_slope, a_offset, a_doDrift, a_doInclineT, ...
   a_adjError, a_adjDate, a_launchDate, a_trajNMeas, a_idMeas, a_decoderId)

% output parameters initialization
o_ok = 0;
o_tabMeas = [];

% current float WMO number
global g_decArgo_floatNum;

% QC flag values (numerical)
global g_decArgo_qcDef;
global g_decArgo_qcNoQc;

% global measurement codes
global g_MC_RPP;

% lists of managed decoders
global g_decArgo_decoderIdListBgcFloatAll;


if (isempty(a_tabMeas.ptsForDoxy))
   if (ismember(a_decoderId, g_decArgo_decoderIdListBgcFloatAll))
      %       if (~ismember(a_tabMeas.measCode, [g_MC_RPP]))
      %          fprintf('INFO: Float #%d Cycle #%d MC %d: Empty ptsForDoxy => not
adjusted\n', ...
      %             g_decArgo_floatNum, ...
      %             a_trajNMeas.outputCycleNumber, ...
      %             a_tabMeas.measCode);
      %       end
      return
   else
      idPres = find(strcmp({a_tabMeas.paramList.name}, 'PRES'));
      idTemp = find(strcmp({a_tabMeas.paramList.name}, 'TEMP'));
      idPsal = find(strcmp({a_tabMeas.paramList.name}, 'PSAL'));
      if (~isempty(idPres) && ~isempty(idTemp) && ~isempty(idPsal))
         a_tabMeas.ptsForDoxy = a_tabMeas.paramData(:, [idPres idTemp idPsal]);
      else
         %          if (~ismember(a_tabMeas.measCode, [g_MC_RPP]))
         %             fprintf('INFO: Float #%d Cycle #%d MC %d: Empty ptsForDoxy => not
adjusted\n', ...
         %                g_decArgo_floatNum, ...
         %                a_trajNMeas.outputCycleNumber, ...
         %                a_tabMeas.measCode);
         %          end
         return
      end
   end
end

% involved parameter information
paramPres = get_netcdf_param_attributes('PRES');
```

```
paramTemp = get_netcdf_param_attributes('TEMP');
paramPsal = get_netcdf_param_attributes('PSAL');
paramDoxy = get_netcdf_param_attributes('DOXY');

% retrieve associated PTS measurements
presValues = a_tabMeas.ptsForDoxy(:, 1);
tempValues = a_tabMeas.ptsForDoxy(:, 2);
psalValues = a_tabMeas.ptsForDoxy(:, 3);

% retrieve DOXY measurements
idDoxy = find(strcmp({a_tabMeas.paramList.name}, 'DOXY'));
doxyValues = a_tabMeas.paramData(:, idDoxy);

% adjust DOXY data
[doxyAdjValues, doxyAdjErrValues] = compute_DOXY_ADJUSTED_traj_meas( ...
   presValues, tempValues, psalValues, doxyValues, ...
   paramPres.fillValue, paramTemp.fillValue, paramPsal.fillValue, paramDoxy.fillValue, ...
   a_doCorPres, a_slope, a_offset, a_doDrift, a_doInclineT, a_launchDate, a_adjError,
a_adjDate, ...
   a_trajNMeas, a_tabMeas, a_idMeas);

if (any(doxyAdjValues ~= paramDoxy.fillValue))

   % create array for adjusted data
   tabMeas = a_tabMeas;
   paramFillValue = get_prof_param_fill_value(tabMeas);
   if (isempty(tabMeas.paramDataAdj))
      tabMeas.paramDataMode = repmat(' ', 1, length(tabMeas.paramList));
      tabMeas.paramDataAdj = repmat(double(paramFillValue), size(tabMeas.paramData, 1), 1);
   end
   if (isempty(tabMeas.paramDataAdjQc))
      tabMeas.paramDataAdjQc = ones(size(tabMeas.paramDataAdj, 1),
length(tabMeas.paramList))*g_decArgo_qcDef;
   end
   if (isempty(tabMeas.paramDataAdjError))
      tabMeas.paramDataAdjError = repmat(double(paramFillValue), size(tabMeas.paramData, 1),
1);
   end

   % store adjusted data
   tabMeas.paramDataMode(idDoxy) = 'A';
   tabMeas.paramDataAdj(:, idDoxy) = doxyAdjValues;

   idNoDef = find(tabMeas.paramDataAdj(:, idDoxy) ~= paramDoxy.fillValue);
   tabMeas.paramDataAdjQc(idNoDef, idDoxy) = g_decArgo_qcNoQc;

   % store error on adjusted data
   if (~isempty(doxyAdjErrValues))
      tabMeas.paramDataAdjError(:, idDoxy) = doxyAdjErrValues;
   end

   % output parameters
   o_ok = 1;
   o_tabMeas = tabMeas;
end

return

% ------------------------------------------------------------------------------
% Adjust DOXY measurements.
% DOXY_ADJUSTED is estimated from an adjustment of PPOX_DOXY at surface on WOA
% climatology.
%
% SYNTAX :
%  [o_DOXY_ADJUSTED, o_DOXY_ADJUSTED_ERROR] = compute_DOXY_ADJUSTED_traj_meas( ...
%    a_PRES, a_TEMP, a_PSAL, a_DOXY, ...
%    a_PRES_fillValue, a_TEMP_fillValue, a_PSAL_fillValue, a_DOXY_fillValue, ...
```

```
%    a_doCorPres, a_slope, a_offset, a_doDrift, a_doInclineT, a_launchDate, a_adjError,
a_adjDate, ...
%    a_trajNMeas, a_tabMeas, a_idMeas)
%
% INPUT PARAMETERS :
%   a_PRES           : input PRES data
%   a_TEMP           : input TEMP data
%   a_PSAL           : input PSAL data
%   a_DOXY           : input DOXY data
%   a_PRES_fillValue : fill value for input PRES data
%   a_TEMP_fillValue : fill value for input TEMP data
%   a_PSAL_fillValue : fill value for input PSAL data
%   a_DOXY_fillValue : fill value for input DOXY data
%   a_DOXY_fillValue : fill value for input DOXY data
%   a_doCorPres      : coefficient for DOXY correction f(PRES) - in CASE 3_2_2 only
%   a_slope          : slope of PPOX_DOXY adjustment
%   a_offset         : slope of PPOX_DOXY adjustment
%   a_doDrift        : drift to be used for PPOX_DOXY adjustment
%   a_doInclineT     : incline_t to be used for PPOX_DOXY adjustment
%   a_launchDate     : float launch date
%   a_adjError       : error on PPOX_DOXY adjusted values
%   a_adjDate        : start date to apply adjustment
%   a_trajNMeas      : measurement structure
%   a_tabMeas        : associated cycle measurements
%   a_idMeas         : index of current measurement in cycle measurements

%
% OUTPUT PARAMETERS :
%   o_DOXY_ADJUSTED       : output DOXY adjusted data
%   o_DOXY_ADJUSTED_ERROR : output error on DOXY adjusted data
%
% EXAMPLES :
%
% SEE ALSO :
% AUTHORS  : Jean-Philippe Rannou (Altran)(jean-philippe.rannou@altran.com)
% ------------------------------------------------------------------------------
% RELEASES :
%   09/01/2021 - RNU - creation
% ------------------------------------------------------------------------------
function [o_DOXY_ADJUSTED, o_DOXY_ADJUSTED_ERROR] = compute_DOXY_ADJUSTED_traj_meas( ...
   a_PRES, a_TEMP, a_PSAL, a_DOXY, ...
   a_PRES_fillValue, a_TEMP_fillValue, a_PSAL_fillValue, a_DOXY_fillValue, ...
   a_doCorPres, a_slope, a_offset, a_doDrift, a_doInclineT, a_launchDate, a_adjError,
a_adjDate, ...
   a_trajNMeas, a_tabMeas, a_idMeas)

% output parameters initialization
o_DOXY_ADJUSTED = ones(length(a_DOXY), 1)*a_DOXY_fillValue;
if (~isnan(a_adjError))
   o_DOXY_ADJUSTED_ERROR = ones(length(a_DOXY), 1)*a_DOXY_fillValue;
else
   o_DOXY_ADJUSTED_ERROR = [];
end

% current float WMO number
global g_decArgo_floatNum;

% retrieve global coefficient default values
global g_decArgo_doxy_202_205_304_d0;
global g_decArgo_doxy_202_205_304_d1;
global g_decArgo_doxy_202_205_304_d2;
global g_decArgo_doxy_202_205_304_d3;
global g_decArgo_doxy_202_205_304_b0;
global g_decArgo_doxy_202_205_304_b1;
global g_decArgo_doxy_202_205_304_b2;
global g_decArgo_doxy_202_205_304_b3;
global g_decArgo_doxy_202_205_304_c0;
```

```
global g_decArgo_doxy_202_205_304_pCoef2;
global g_decArgo_doxy_202_205_304_pCoef3;

% global default values
global g_decArgo_ncDateDef;


if (isempty(a_PRES) || isempty(a_TEMP) || isempty(a_PSAL) || isempty(a_DOXY))
   return
end

idDef = find( ...
   (a_PRES == a_PRES_fillValue) | ...
   (a_TEMP == a_TEMP_fillValue) | ...
   (a_PSAL == a_PSAL_fillValue) | ...
   (a_DOXY == a_DOXY_fillValue));
idNoDef = setdiff(1:length(a_DOXY), idDef);

if (~isempty(idNoDef))

   presValues = a_PRES(idNoDef);
   tempValues = a_TEMP(idNoDef);
   psalValues = a_PSAL(idNoDef);
   doxyValues = a_DOXY(idNoDef);

   % correction of DOXY due to pressure (in CASE 3_2_2 only)
   if (~isnan(a_doCorPres))
      doxyValues = doxyValues .* (1 + a_doCorPres*presValues/1000);
   end

   % convert DOXY into DOXY_in_molar_units
   % units convertion (micromol/kg to micromol/L)
   [measLon, measLat] = get_meas_location(a_trajNMeas.cycleNumber, a_trajNMeas.profileNumber,
[]);
   rho = potential_density_gsw(presValues, tempValues, psalValues, 0, measLon, measLat);
   rho = rho/1000;
   molarDoxyValues = doxyValues .* rho;

   % pressure effect un-correction:
   % at presValue, optode quenched by different pO2 inside membrane than pO2
   % outside in seawater due to re-equilibration effect
   % translate already corrected value (outside conditions) back to sensed value
   % (inside membrane)
   oxygenPresUncomp = calcoxy_presuncomp(molarDoxyValues, presValues, tempValues, ...
      g_decArgo_doxy_202_205_304_pCoef2, ...
      g_decArgo_doxy_202_205_304_pCoef3 ...
      );

   % convert DOXY_in_molar_units_and_inside_conditions into PPOX_DOXY
   % units convertion (micromol/L to hPa)
   ppoxDoxyValues = O2ctoO2p(oxygenPresUncomp, tempValues, psalValues, presValues, ...
      g_decArgo_doxy_202_205_304_d0, ...
      g_decArgo_doxy_202_205_304_d1, ...
      g_decArgo_doxy_202_205_304_d2, ...
      g_decArgo_doxy_202_205_304_d3, ...
      g_decArgo_doxy_202_205_304_b0, ...
      g_decArgo_doxy_202_205_304_b1, ...
      g_decArgo_doxy_202_205_304_b2, ...
      g_decArgo_doxy_202_205_304_b3, ...
      g_decArgo_doxy_202_205_304_c0 ...
      );

   % adjust PPOX_DOXY
   measDate = '';
   if (~isempty(a_tabMeas.juldAdj) && (a_tabMeas.juldAdj ~= g_decArgo_ncDateDef))
      measDate = a_tabMeas.juldAdj;
   elseif (~isempty(a_tabMeas.juld) && (a_tabMeas.juld ~= g_decArgo_ncDateDef))
```

```
      measDate = a_tabMeas.juld;
   else
      for idM = 1:length(a_trajNMeas.tabMeas)
         prevId = a_idMeas - idM;
         nextId = a_idMeas + idM;
         if ((prevId > 0) && (nextId <= length(a_trajNMeas.tabMeas)))
            measDatePrev = '';
            if (~isempty(a_trajNMeas.tabMeas(prevId).juldAdj) &&
(a_trajNMeas.tabMeas(prevId).juldAdj ~= g_decArgo_ncDateDef))
               measDatePrev = a_trajNMeas.tabMeas(prevId).juldAdj;
            elseif (~isempty(a_trajNMeas.tabMeas(prevId).juld) &&
(a_trajNMeas.tabMeas(prevId).juld ~= g_decArgo_ncDateDef))
               measDatePrev = a_trajNMeas.tabMeas(prevId).juld;
            end
            measDateNext = '';
            if (~isempty(a_trajNMeas.tabMeas(nextId).juldAdj) &&
(a_trajNMeas.tabMeas(nextId).juldAdj ~= g_decArgo_ncDateDef))
               measDateNext = a_trajNMeas.tabMeas(nextId).juldAdj;
            elseif (~isempty(a_trajNMeas.tabMeas(nextId).juld) &&
(a_trajNMeas.tabMeas(nextId).juld ~= g_decArgo_ncDateDef))
               measDateNext = a_trajNMeas.tabMeas(nextId).juld;
            end
            if (~isempty(measDatePrev) && ~isempty(measDateNext))
               measDate = measDatePrev + (measDateNext-measDatePrev)/2;
               break
            elseif (~isempty(measDatePrev))
               measDate = measDatePrev;
               break
            elseif (~isempty(measDateNext))
               measDate = measDateNext;
               break
            end
         elseif (prevId > 0)
            measDatePrev = '';
            if (~isempty(a_trajNMeas.tabMeas(prevId).juldAdj) &&
(a_trajNMeas.tabMeas(prevId).juldAdj ~= g_decArgo_ncDateDef))
               measDatePrev = a_trajNMeas.tabMeas(prevId).juldAdj;
            elseif (~isempty(a_trajNMeas.tabMeas(prevId).juld) &&
(a_trajNMeas.tabMeas(prevId).juld ~= g_decArgo_ncDateDef))
               measDatePrev = a_trajNMeas.tabMeas(prevId).juld;
            end
            if (~isempty(measDatePrev))
               measDate = measDatePrev;
               break
            end
         elseif (nextId <= length(a_trajNMeas.tabMeas))
            measDateNext = '';
            if (~isempty(a_trajNMeas.tabMeas(nextId).juldAdj) &&
(a_trajNMeas.tabMeas(nextId).juldAdj ~= g_decArgo_ncDateDef))
               measDateNext = a_trajNMeas.tabMeas(nextId).juldAdj;
            elseif (~isempty(a_trajNMeas.tabMeas(nextId).juld) &&
(a_trajNMeas.tabMeas(nextId).juld ~= g_decArgo_ncDateDef))
               measDateNext = a_trajNMeas.tabMeas(nextId).juld;
            end
            if (~isempty(measDateNext))
               measDate = measDateNext;
               break
            end
         else
            break
         end
      end
   end

   if (~isempty(measDate))
      ppoxDoxyAdjValues = (a_slope * (1 + a_doDrift/100 * (measDate-a_launchDate)/365) +
a_doInclineT*tempValues) .* (ppoxDoxyValues + a_offset);
```

```
   else
      fprintf('WARNING: Float #%d Cycle #%d: DOXY measurement is not dated - DOXY_ADJUSTED set
to FillValue\n', ...
         g_decArgo_floatNum, ...
         a_trajNMeas.outputCycleNumber);
      return
   end

   % convert PPOX_ADJUSTED into DOXY_ADJUSTED_in_molar_units_and_inside_conditions
   % units convertion (hPa to micromol/L)
   oxygenAdjPresUncomp = O2ptoO2c(ppoxDoxyAdjValues, tempValues, psalValues, presValues, ...
      g_decArgo_doxy_202_205_304_d0, ...
      g_decArgo_doxy_202_205_304_d1, ...
      g_decArgo_doxy_202_205_304_d2, ...
      g_decArgo_doxy_202_205_304_d3, ...
      g_decArgo_doxy_202_205_304_b0, ...
      g_decArgo_doxy_202_205_304_b1, ...
      g_decArgo_doxy_202_205_304_b2, ...
      g_decArgo_doxy_202_205_304_b3, ...
      g_decArgo_doxy_202_205_304_c0 ...
      );

   % pressure effect re-correction:
   % at presValue, optode quenched by different pO2 inside membrane than pO2
   % outside in seawater due to re-equilibration effect
   % translate adjusted sensed value (inside membrane) to adjusted corrected
   % value (outside conditions)
   molarDoxyAdjValues  = calcoxy_prescomp(oxygenAdjPresUncomp, presValues, tempValues, ...
      g_decArgo_doxy_202_205_304_pCoef2, ...
      g_decArgo_doxy_202_205_304_pCoef3 ...
      );

   % convert DOXY_ADJUSTED_in_molar_units into DOXY_ADJUSTED
   % units convertion (micromol/L to micromol/kg)
   doxyAdjValues = molarDoxyAdjValues ./ rho;

   o_DOXY_ADJUSTED(idNoDef) = doxyAdjValues;

   % compute DOXY_ADJUSTED_ERROR

   if (~isnan(a_adjError))

      % use PPOX_DOXY_ADJUSTED_ERROR from META-DATA
      ppoxDoxyAdjErrValues = a_adjError;

      % increase PPOX_DOXY_ADJUSTED_ERROR with time (1 mbar/year)
      if (~isempty(a_adjDate))
         ppoxDoxyAdjErrValues = ppoxDoxyAdjErrValues + (measDate - a_adjDate)/365;
      end

      % convert PPOX_ADJUSTED_ERROR into
DOXY_ADJUSTED_ERROR_in_molar_units_and_inside_conditions
      % units convertion (hPa to micromol/L)
      oxygenAdjErrPresUncomp = O2ptoO2c(ppoxDoxyAdjErrValues, tempValues, psalValues,
presValues, ...
         g_decArgo_doxy_202_205_304_d0, ...
         g_decArgo_doxy_202_205_304_d1, ...
         g_decArgo_doxy_202_205_304_d2, ...
         g_decArgo_doxy_202_205_304_d3, ...
         g_decArgo_doxy_202_205_304_b0, ...
         g_decArgo_doxy_202_205_304_b1, ...
         g_decArgo_doxy_202_205_304_b2, ...
         g_decArgo_doxy_202_205_304_b3, ...
         g_decArgo_doxy_202_205_304_c0 ...
         );

      % pressure effect re-correction:
```

```
      % at presValue, optode quenched by different pO2 inside membrane than pO2
      % outside in seawater due to re-equilibration effect
      % translate adjusted sensed value (inside membrane) to adjusted corrected
      % value (outside conditions)
      molarDoxyAdjErrValues  = calcoxy_prescomp(oxygenAdjErrPresUncomp, presValues,
tempValues, ...
         g_decArgo_doxy_202_205_304_pCoef2, ...
         g_decArgo_doxy_202_205_304_pCoef3 ...
         );

      % convert DOXY_ADJUSTED_ERROR_in_molar_units into DOXY_ADJUSTED_ERROR
      % units convertion (micromol/L to micromol/kg)
      doxyAdjErrValues = molarDoxyAdjErrValues ./ rho;

      o_DOXY_ADJUSTED_ERROR(idNoDef) = doxyAdjErrValues;
   end
end

return

% ------------------------------------------------------------------------------
% Perform real time adjustment on one DOXY profile data.
%
% SYNTAX :
%  [o_ok, o_profile] = adjust_doxy_profile( ...
%    a_profile, a_tabProfiles, ...
%    a_doCorPres, a_slope, a_offset, a_doDrift, a_doInclineT, ...
%    a_adjError, a_adjDate, a_launchDate, a_decoderId)
%
% INPUT PARAMETERS :
%   a_profile     : input DOXY profile structure
%   a_tabProfiles : profile structures with the same cycle number and
%                   direction as the DOXY one
%   a_doCorPres   : coefficient for DOXY correction f(PRES) - in CASE 3_2_2 only
%   a_slope       : slope to be used for PPOX_DOXY adjustment
%   a_offset      : offset to be used for PPOX_DOXY adjustment
%   a_doDrift     : drift to be used for PPOX_DOXY adjustment
%   a_doInclineT  : incline_t to be used for PPOX_DOXY adjustment
%   a_adjError    : error on PPOX_DOXY adjusted values
%   a_adjDate     : start date to apply adjustment
%   a_launchDate  : float launch date
%   a_decoderId   : float decoder Id
%
% OUTPUT PARAMETERS :
%   o_ok      : 1 if the adjustment has been performed, 0 otherwise
%   o_profile : output DOXY profile structure
%
% EXAMPLES :
%
% SEE ALSO :
% AUTHORS  : Jean-Philippe Rannou (Altran)(jean-philippe.rannou@altran.com)
% ------------------------------------------------------------------------------
% RELEASES :
%   07/03/2019 - RNU - creation
% ------------------------------------------------------------------------------
function [o_ok, o_profile] = adjust_doxy_profile( ...
   a_profile, a_tabProfiles, ...
   a_doCorPres, a_slope, a_offset, a_doDrift, a_doInclineT, ...
   a_adjError, a_adjDate, a_launchDate, a_decoderId)

% output parameters initialization
o_ok = 0;
o_profile = [];

% current float WMO number
global g_decArgo_floatNum;
```

```
% QC flag values (numerical)
global g_decArgo_qcDef;
global g_decArgo_qcNoQc;

% lists of managed decoders
global g_decArgo_decoderIdListBgcFloatAll;


% involved parameter information
paramPres = get_netcdf_param_attributes('PRES');
paramTemp = get_netcdf_param_attributes('TEMP');
paramPsal = get_netcdf_param_attributes('PSAL');
paramDoxy = get_netcdf_param_attributes('DOXY');

% retrieve or interpolate PTS measurements
presValues = [];
tempValues = [];
psalValues = [];
doxyValues = [];
idPres = find(strcmp({a_profile.paramList.name}, 'PRES'));
idTemp = find(strcmp({a_profile.paramList.name}, 'TEMP'));
idPsal = find(strcmp({a_profile.paramList.name}, 'PSAL'));
idDoxy = find(strcmp({a_profile.paramList.name}, 'DOXY'));

if (~ismember(a_decoderId, g_decArgo_decoderIdListBgcFloatAll))

   % case of a PTSO float
   presValues = a_profile.data(:, idPres);
   tempValues = a_profile.data(:, idTemp);
   psalValues = a_profile.data(:, idPsal);
   doxyValues = a_profile.data(:, idDoxy);
else

   % case of a 'real' BGC float

   % create a PTS profile by concatenating the near-surface and the primary
   % sampling profiles
   idNssProf = [];
   idPsProf = [];
   for idProf = 1:length(a_tabProfiles)
      profile = a_tabProfiles(idProf);
      if (strncmp(profile.vertSamplingScheme, 'Near-surface sampling:', length('Near-surface
sampling:')))
         idNssPres = find(strcmp({profile.paramList.name}, 'PRES'));
         idNssTemp = find(strcmp({profile.paramList.name}, 'TEMP'));
         idNssPsal = find(strcmp({profile.paramList.name}, 'PSAL'));
         if (~isempty(idNssPres) && ~isempty(idNssTemp) && ~isempty(idNssPsal))
            idNssProf = idProf;
         end
      elseif (strncmp(profile.vertSamplingScheme, 'Primary sampling:', length('Primary
sampling:')))
         idPsPres = find(strcmp({profile.paramList.name}, 'PRES'));
         idPsTemp = find(strcmp({profile.paramList.name}, 'TEMP'));
         idPsPsal = find(strcmp({profile.paramList.name}, 'PSAL'));
         if (~isempty(idPsPres) && ~isempty(idPsTemp) && ~isempty(idPsPsal))
            idPsProf = idProf;
         end
      end
      if (~isempty(idNssProf) && ~isempty(idPsProf))
         break
      end
   end

   if (~isempty(idNssProf) && ~isempty(idPsProf))
      ctdPresData = [a_tabProfiles(idPsProf).data(:, idPsPres);
a_tabProfiles(idNssProf).data(:, idNssPres)];
```

```
      ctdTempData = [a_tabProfiles(idPsProf).data(:, idPsTemp);
a_tabProfiles(idNssProf).data(:, idNssTemp)];
      ctdPsalData = [a_tabProfiles(idPsProf).data(:, idPsPsal);
a_tabProfiles(idNssProf).data(:, idNssPsal)];
   elseif (~isempty(idPsProf))
      ctdPresData = a_tabProfiles(idPsProf).data(:, idPsPres);
      ctdTempData = a_tabProfiles(idPsProf).data(:, idPsTemp);
      ctdPsalData = a_tabProfiles(idPsProf).data(:, idPsPsal);
   elseif (~isempty(idNssProf))
      ctdPresData = a_tabProfiles(idNssProf).data(:, idNssPres);
      ctdTempData = a_tabProfiles(idNssProf).data(:, idNssTemp);
      ctdPsalData = a_tabProfiles(idNssProf).data(:, idNssPsal);
   else
      ctdPresData = [];
      ctdTempData = [];
      ctdPsalData = [];
   end

   % clean fill values
   idNoDefPts = find((ctdPresData ~= paramPres.fillValue) & ...
      (ctdTempData ~= paramTemp.fillValue) & ...
      (ctdPsalData ~= paramPsal.fillValue));

   ctdPresData = ctdPresData(idNoDefPts);
   ctdTempData = ctdTempData(idNoDefPts);
   ctdPsalData = ctdPsalData(idNoDefPts);

   if (~isempty(ctdPresData))

      % interpolate and extrapolate the PTS data at the pressures of the
      % DOXY measurements
      ctdIntData = compute_interpolated_CTD_measurements(...
         [ctdPresData ctdTempData ctdPsalData], a_profile.data(:, idPres),
a_profile.direction);

      presValues = ctdIntData(:, 1);
      tempValues = ctdIntData(:, 2);
      psalValues = ctdIntData(:, 3);
      doxyValues = a_profile.data(:, idDoxy);
   else
      fprintf('WARNING: Float #%d Cycle #%d%c: unable to find the associated CTD profile to
adjust DOXY parameter - DOXY data cannot be adjusted\n', ...
         g_decArgo_floatNum, ...
         a_profile.outputCycleNumber, a_profile.direction);
   end
end

if (~isempty(presValues))

   % adjust DOXY data
   [doxyAdjValues, doxyAdjErrValues] = compute_DOXY_ADJUSTED_profile( ...
      presValues, tempValues, psalValues, doxyValues, ...
      paramPres.fillValue, paramTemp.fillValue, paramPsal.fillValue, paramDoxy.fillValue, ...
      a_doCorPres, a_slope, a_offset, a_doDrift, a_doInclineT, a_launchDate, a_adjError,
a_adjDate, a_profile);

   if (any(doxyAdjValues ~= paramDoxy.fillValue))

      % create array for adjusted data
      profile = a_profile;
      paramFillValue = get_prof_param_fill_value(profile);
      if (isempty(profile.dataAdj))
         profile.paramDataMode = repmat(' ', 1, length(profile.paramList));
         profile.dataAdj = repmat(double(paramFillValue), size(profile.data, 1), 1);
      end
      if (isempty(profile.dataAdjQc))
```

```
         profile.dataAdjQc = ones(size(profile.dataAdj, 1),
length(profile.paramList))*g_decArgo_qcDef;
      end
      if (isempty(profile.dataAdjError) && ~isempty(doxyAdjErrValues))
         profile.dataAdjError = repmat(double(paramFillValue), size(profile.data, 1), 1);
      end

      % store adjusted data
      profile.paramDataMode(idDoxy) = 'A';
      profile.dataAdj(:, idDoxy) = doxyAdjValues;
      idNoDef = find(doxyAdjValues ~= paramDoxy.fillValue);
      profile.dataAdjQc(idNoDef, idDoxy) = g_decArgo_qcNoQc;
      if (~isempty(doxyAdjErrValues))
         profile.dataAdjError(:, idDoxy) = doxyAdjErrValues;
      end

      % output parameters
      o_ok = 1;
      o_profile = profile;
   end
end

return

% ------------------------------------------------------------------------------
% Interpolate the T and S measurements of a CTD profile at given P levels.
%
% SYNTAX :
%  [o_ctdIntData] = compute_interpolated_CTD_measurements( ...
%    a_ctdMeasData, a_presData, a_presData, a_profDir)
%
% INPUT PARAMETERS :
%   a_ctdMeasData : CTD profile measurements
%   a_presData    : P levels of T and S measurement interpolation
%   a_profDir     : profile direction
%
% OUTPUT PARAMETERS :
%   o_ctdIntData : CTD interpolated data
%
% EXAMPLES :
%
% SEE ALSO :
% AUTHORS  : Jean-Philippe Rannou (Altran)(jean-philippe.rannou@altran.com)
% ------------------------------------------------------------------------------
% RELEASES :
%   06/02/2014 - RNU - creation
% ------------------------------------------------------------------------------
function [o_ctdIntData] = compute_interpolated_CTD_measurements( ...
   a_ctdMeasData, a_presData, a_profDir)

% output parameters initialization
o_ctdIntData = [];


if (isempty(a_ctdMeasData))
   return
end

paramPres = get_netcdf_param_attributes('PRES');
paramTemp = get_netcdf_param_attributes('TEMP');
paramSal = get_netcdf_param_attributes('PSAL');

% get the measurement levels of output data
idNoDefOutput = find((a_presData ~= paramPres.fillValue));

% interpolate the T and S measurements at the output P levels
idNoDefInput = find(~((a_ctdMeasData(:, 1) == paramPres.fillValue) | ...
```

```
   (a_ctdMeasData(:, 2) == paramTemp.fillValue) | ...
   (a_ctdMeasData(:, 3) == paramSal.fillValue)));

if (~isempty(idNoDefInput))

   % get PTS measurements
   ctdPresData = a_ctdMeasData(idNoDefInput, 1);
   ctdTempData = a_ctdMeasData(idNoDefInput, 2);
   ctdPsalData = a_ctdMeasData(idNoDefInput, 3);

   % if it is a ascending profile, flip measurements up to down
   %    if (length(find(diff(ctdPresData)<0)) > length(ctdPresData)/2)
   if (a_profDir == 'A')
      ctdPresData = flipud(ctdPresData);
      ctdTempData = flipud(ctdTempData);
      ctdPsalData = flipud(ctdPsalData);
   end

   if (length(ctdPresData) > 1)

      % consider increasing pressures only (we start the algorithm from the middle
      % of the profile)
      idToDelete = [];
      idStart = fix(length(ctdPresData)/2);
      pMin = ctdPresData(idStart);
      for id = idStart-1:-1:1
         if (ctdPresData(id) >= pMin)
            idToDelete = [idToDelete id];
         else
            pMin = ctdPresData(id);
         end
      end
      pMax = ctdPresData(idStart);
      for id = idStart+1:length(ctdPresData)
         if (ctdPresData(id) <= pMax)
            idToDelete = [idToDelete id];
         else
            pMax = ctdPresData(id);
         end
      end

      ctdPresData(idToDelete) = [];
      ctdTempData(idToDelete) = [];
      ctdPsalData(idToDelete) = [];
   end

   if (~isempty(ctdPresData))

      % duplicate T&S values 10 dbar above the shallowest level
      ctdPresData = [ctdPresData(1)-10; ctdPresData];
      ctdTempData = [ctdTempData(1); ctdTempData];
      ctdPsalData = [ctdPsalData(1); ctdPsalData];

      % duplicate T&S values 50 dbar below the deepest level
      ctdPresData = [ctdPresData; ctdPresData(end)+50];
      ctdTempData = [ctdTempData; ctdTempData(end)];
      ctdPsalData = [ctdPsalData; ctdPsalData(end)];

      tempIntData = interp1(ctdPresData, ...
         ctdTempData, ...
         a_presData(idNoDefOutput), 'linear');
      psalIntData = interp1(ctdPresData, ...
         ctdPsalData, ...
         a_presData(idNoDefOutput), 'linear');

      tempIntData(isnan(tempIntData)) = paramTemp.fillValue;
      psalIntData(isnan(psalIntData)) = paramSal.fillValue;
```

```
      % output parameters
      o_ctdIntData = [ ...
         a_presData ...
         ones(length(a_presData), 1)*paramTemp.fillValue ...
         ones(length(a_presData), 1)*paramSal.fillValue];
      o_ctdIntData(idNoDefOutput, 2) = tempIntData;
      o_ctdIntData(idNoDefOutput, 3) = psalIntData;
   end
end

return

% ------------------------------------------------------------------------------
% Adjust profile DOXY measurements.
% DOXY_ADJUSTED is estimated from an adjustment of PPOX_DOXY at surface on WOA
% climatology.
%
% SYNTAX :
%  [o_DOXY_ADJUSTED, o_DOXY_ADJUSTED_ERROR] = compute_DOXY_ADJUSTED_profile( ...
%    a_PRES, a_TEMP, a_PSAL, a_DOXY, ...
%    a_PRES_fillValue, a_TEMP_fillValue, a_PSAL_fillValue, a_DOXY_fillValue, ...
%    a_doCorPres, a_slope, a_offset, a_doDrift, a_doInclineT, a_launchDate, a_adjError,
a_adjDate, a_profOptode)
%
% INPUT PARAMETERS :
%   a_PRES           : input PRES data
%   a_TEMP           : input TEMP data
%   a_PSAL           : input PSAL data
%   a_DOXY           : input DOXY data
%   a_PRES_fillValue : fill value for input PRES data
%   a_TEMP_fillValue : fill value for input TEMP data
%   a_PSAL_fillValue : fill value for input PSAL data
%   a_DOXY_fillValue : fill value for input DOXY data
%   a_DOXY_fillValue : fill value for input DOXY data
%   a_doCorPres      : coefficient for DOXY correction f(PRES) - in CASE 3_2_2 only
%   a_slope          : slope of PPOX_DOXY adjustment
%   a_offset         : slope of PPOX_DOXY adjustment
%   a_doDrift        : drift to be used for PPOX_DOXY adjustment
%   a_doInclineT     : incline_t to be used for PPOX_DOXY adjustment
%   a_launchDate     : float launch date
%   a_adjError       : error on PPOX_DOXY adjusted values
%   a_adjDate        : start date to apply adjustment
%   a_profOptode     : OPTODE profile structure
%
% OUTPUT PARAMETERS :
%   o_DOXY_ADJUSTED       : output DOXY adjusted data
%   o_DOXY_ADJUSTED_ERROR : output error on DOXY adjusted data
%
% EXAMPLES :
%
% SEE ALSO :
% AUTHORS  : Jean-Philippe Rannou (Altran)(jean-philippe.rannou@altran.com)
% ------------------------------------------------------------------------------
% RELEASES :
%   07/04/2019 - RNU - creation
% ------------------------------------------------------------------------------
function [o_DOXY_ADJUSTED, o_DOXY_ADJUSTED_ERROR] = compute_DOXY_ADJUSTED_profile( ...
   a_PRES, a_TEMP, a_PSAL, a_DOXY, ...
   a_PRES_fillValue, a_TEMP_fillValue, a_PSAL_fillValue, a_DOXY_fillValue, ...
   a_doCorPres, a_slope, a_offset, a_doDrift, a_doInclineT, a_launchDate, a_adjError,
a_adjDate, a_profOptode)

% output parameters initialization
o_DOXY_ADJUSTED = ones(length(a_DOXY), 1)*a_DOXY_fillValue;
if (~isnan(a_adjError))
   o_DOXY_ADJUSTED_ERROR = ones(length(a_DOXY), 1)*a_DOXY_fillValue;
```

```
else
   o_DOXY_ADJUSTED_ERROR = [];
end

% current float WMO number
global g_decArgo_floatNum;

% retrieve global coefficient default values
global g_decArgo_doxy_202_205_304_d0;
global g_decArgo_doxy_202_205_304_d1;
global g_decArgo_doxy_202_205_304_d2;
global g_decArgo_doxy_202_205_304_d3;
global g_decArgo_doxy_202_205_304_b0;
global g_decArgo_doxy_202_205_304_b1;
global g_decArgo_doxy_202_205_304_b2;
global g_decArgo_doxy_202_205_304_b3;
global g_decArgo_doxy_202_205_304_c0;
global g_decArgo_doxy_202_205_304_pCoef2;
global g_decArgo_doxy_202_205_304_pCoef3;

% global default values
global g_decArgo_dateDef;


if (isempty(a_PRES) || isempty(a_TEMP) || isempty(a_PSAL) || isempty(a_DOXY))
   return
end

idDef = find( ...
   (a_PRES == a_PRES_fillValue) | ...
   (a_TEMP == a_TEMP_fillValue) | ...
   (a_PSAL == a_PSAL_fillValue) | ...
   (a_DOXY == a_DOXY_fillValue));
idNoDef = setdiff(1:length(a_DOXY), idDef);

if (~isempty(idNoDef))

   presValues = a_PRES(idNoDef);
   tempValues = a_TEMP(idNoDef);
   psalValues = a_PSAL(idNoDef);
   doxyValues = a_DOXY(idNoDef);

   % correction of DOXY due to pressure (in CASE 3_2_2 only)
   if (~isnan(a_doCorPres))
      doxyValues = doxyValues .* (1 + a_doCorPres*presValues/1000);
   end

   % convert DOXY into DOXY_in_molar_units
   % units convertion (micromol/kg to micromol/L)
   [measLon, measLat] = get_meas_location(a_profOptode.cycleNumber,
a_profOptode.profileNumber, a_profOptode);
   rho = potential_density_gsw(presValues, tempValues, psalValues, 0, measLon, measLat);
   rho = rho/1000;
   molarDoxyValues = doxyValues .* rho;

   % pressure effect un-correction:
   % at presValue, optode quenched by different pO2 inside membrane than pO2
   % outside in seawater due to re-equilibration effect
   % translate already corrected value (outside conditions) back to sensed value
   % (inside membrane)
   oxygenPresUncomp = calcoxy_presuncomp(molarDoxyValues, presValues, tempValues, ...
      g_decArgo_doxy_202_205_304_pCoef2, ...
      g_decArgo_doxy_202_205_304_pCoef3 ...
      );

   % convert DOXY_in_molar_units_and_inside_conditions into PPOX_DOXY
   % units convertion (micromol/L to hPa)
```

```
   ppoxDoxyValues = O2ctoO2p(oxygenPresUncomp, tempValues, psalValues, presValues, ...
      g_decArgo_doxy_202_205_304_d0, ...
      g_decArgo_doxy_202_205_304_d1, ...
      g_decArgo_doxy_202_205_304_d2, ...
      g_decArgo_doxy_202_205_304_d3, ...
      g_decArgo_doxy_202_205_304_b0, ...
      g_decArgo_doxy_202_205_304_b1, ...
      g_decArgo_doxy_202_205_304_b2, ...
      g_decArgo_doxy_202_205_304_b3, ...
      g_decArgo_doxy_202_205_304_c0 ...
      );

   % adjust PPOX_DOXY
   if (a_profOptode.date ~= g_decArgo_dateDef)
      ppoxDoxyAdjValues = (a_slope * (1 + a_doDrift/100 * (a_profOptode.date-
a_launchDate)/365) + a_doInclineT*tempValues) .* (ppoxDoxyValues + a_offset);
   else
      fprintf('WARNING: Float #%d Cycle #%d%c: profile is not dated - DOXY_ADJUSTED set to
FillValue\n', ...
         g_decArgo_floatNum, ...
         a_profOptode.outputCycleNumber, a_profOptode.direction);
      return
   end

   % convert PPOX_ADJUSTED into DOXY_ADJUSTED_in_molar_units_and_inside_conditions
   % units convertion (hPa to micromol/L)
   oxygenAdjPresUncomp = O2ptoO2c(ppoxDoxyAdjValues, tempValues, psalValues, presValues, ...
      g_decArgo_doxy_202_205_304_d0, ...
      g_decArgo_doxy_202_205_304_d1, ...
      g_decArgo_doxy_202_205_304_d2, ...
      g_decArgo_doxy_202_205_304_d3, ...
      g_decArgo_doxy_202_205_304_b0, ...
      g_decArgo_doxy_202_205_304_b1, ...
      g_decArgo_doxy_202_205_304_b2, ...
      g_decArgo_doxy_202_205_304_b3, ...
      g_decArgo_doxy_202_205_304_c0 ...
      );

   % pressure effect re-correction:
   % at presValue, optode quenched by different pO2 inside membrane than pO2
   % outside in seawater due to re-equilibration effect
   % translate adjusted sensed value (inside membrane) to adjusted corrected
   % value (outside conditions)
   molarDoxyAdjValues  = calcoxy_prescomp(oxygenAdjPresUncomp, presValues, tempValues, ...
      g_decArgo_doxy_202_205_304_pCoef2, ...
      g_decArgo_doxy_202_205_304_pCoef3 ...
      );

   % convert DOXY_ADJUSTED_in_molar_units into DOXY_ADJUSTED
   % units convertion (micromol/L to micromol/kg)
   doxyAdjValues = molarDoxyAdjValues ./ rho;

   o_DOXY_ADJUSTED(idNoDef) = doxyAdjValues;

   % compute DOXY_ADJUSTED_ERROR

   if (~isnan(a_adjError))

      % use PPOX_DOXY_ADJUSTED_ERROR from META-DATA
      ppoxDoxyAdjErrValues = a_adjError;

      % increase PPOX_DOXY_ADJUSTED_ERROR with time (1 mbar/year)
      if (~isempty(a_adjDate))
         ppoxDoxyAdjErrValues = ppoxDoxyAdjErrValues + (a_profOptode.date - a_adjDate)/365;
      end
```

```
      % convert PPOX_ADJUSTED_ERROR into
DOXY_ADJUSTED_ERROR_in_molar_units_and_inside_conditions
      % units convertion (hPa to micromol/L)
      oxygenAdjErrPresUncomp = O2ptoO2c(ppoxDoxyAdjErrValues, tempValues, psalValues,
presValues, ...
         g_decArgo_doxy_202_205_304_d0, ...
         g_decArgo_doxy_202_205_304_d1, ...
         g_decArgo_doxy_202_205_304_d2, ...
         g_decArgo_doxy_202_205_304_d3, ...
         g_decArgo_doxy_202_205_304_b0, ...
         g_decArgo_doxy_202_205_304_b1, ...
         g_decArgo_doxy_202_205_304_b2, ...
         g_decArgo_doxy_202_205_304_b3, ...
         g_decArgo_doxy_202_205_304_c0 ...
         );

      % pressure effect re-correction:
      % at presValue, optode quenched by different pO2 inside membrane than pO2
      % outside in seawater due to re-equilibration effect
      % translate adjusted sensed value (inside membrane) to adjusted corrected
      % value (outside conditions)
      molarDoxyAdjErrValues  = calcoxy_prescomp(oxygenAdjErrPresUncomp, presValues,
tempValues, ...
         g_decArgo_doxy_202_205_304_pCoef2, ...
         g_decArgo_doxy_202_205_304_pCoef3 ...
         );

      % convert DOXY_ADJUSTED_ERROR_in_molar_units into DOXY_ADJUSTED_ERROR
      % units convertion (micromol/L to micromol/kg)
      doxyAdjErrValues = molarDoxyAdjErrValues ./ rho;

      o_DOXY_ADJUSTED_ERROR(idNoDef) = doxyAdjErrValues;
   end
end

return

% ------------------------------------------------------------------------------
% Undo Correct DO (in micromol/L) from pressure effect.
%
% SYNTAX :
%  [o_oxygen] = calcoxy_presuncomp(a_oxygenPrescomp, a_pres, a_temp, ...
%    a_pCoef2, a_pCoef3)
%
% INPUT PARAMETERS :
%   o_oxygenPrescomp      : DO values (in micromol/L) corrected from pressure effect
%   a_pres                : PRES values
%   a_temp                : TEMP values
%   a_pCoef2 and a_pCoef3 : additional coefficient values
%
% OUTPUT PARAMETERS :
%   a_oxygen              : DO values
%
% EXAMPLES :
%
% SEE ALSO :
% AUTHORS  : Jean-Philippe Rannou (Altran)(jean-philippe.rannou@altran.com)
% ------------------------------------------------------------------------------
% RELEASES :
%   05/20/2011 - Virginie THIERRY - creation
%   05/17/2016 - RNU - update
% ------------------------------------------------------------------------------
function [o_oxygen] = calcoxy_presuncomp(a_oxygenPrescomp, a_pres, a_temp, ...
   a_pCoef2, a_pCoef3)

% pressure compensation correction
o_oxygen = a_oxygenPrescomp ./ (1 + ((a_pCoef2 .* a_temp) + a_pCoef3) .* a_pres/1000);
```

```
return

% ------------------------------------------------------------------------------
% Correct DO (in micromol/L) from pressure effect.
%
% SYNTAX :
%  [o_oxygenPrescomp] = calcoxy_prescomp(a_oxygen, a_pres, a_temp, ...
%    a_pCoef2, a_pCoef3)
%
% INPUT PARAMETERS :
%   a_oxygen              : DO values
%   a_pres                : PRES values
%   a_temp                : TEMP values
%   a_pCoef2 and a_pCoef3 : additional coefficient values
%
% OUTPUT PARAMETERS :
%   o_oxygenPrescomp : DO values (in micromol/L) corrected from pressure effect
%
% EXAMPLES :
%
% SEE ALSO :
% AUTHORS  : Jean-Philippe Rannou (Altran)(jean-philippe.rannou@altran.com)
% ------------------------------------------------------------------------------
% RELEASES :
%   05/20/2011 - Virginie THIERRY - creation
%   05/17/2016 - RNU - update
% ------------------------------------------------------------------------------
function [o_oxygenPrescomp] = calcoxy_prescomp(a_oxygen, a_pres, a_temp, ...
   a_pCoef2, a_pCoef3)

% pressure compensation correction
o_oxygenPrescomp = a_oxygen .* (1 + ((a_pCoef2 .* a_temp) + a_pCoef3) .* a_pres/1000);

return
```