Section 6: Data Processing Modules

All data processing is performed on converted data from a .cnv file.

Module Name	Module Description
Align CTD	Align data relative to pressure (typically used for conductivity, temperature, and oxygen).
Bin Average	Average data, basing bins on pressure, depth, scan number, or time range.
Buoyancy	Compute Brunt Väisälä buoyancy and stability frequency.
Cell Thermal Mass	Perform conductivity thermal mass correction.
Derive	Calculate salinity, density, sound velocity, oxygen, potential temperature, dynamic height, etc. based on EOS-80 (Practical Salinity) equations.
Derive TEOS-10	Calculate thermodynamic properties based on TEOS-10 (Absolute Salinity).
Filter	Low-pass filter columns of data.
Loop Edit	Mark a scan with *badflag* if scan fails pressure reversal or minimum velocity tests.
Wild Edit	Mark a data value with *badflag* to eliminate wild points.
Window Filter	Filter data with triangle, cosine, boxcar, Gaussian, or median window.

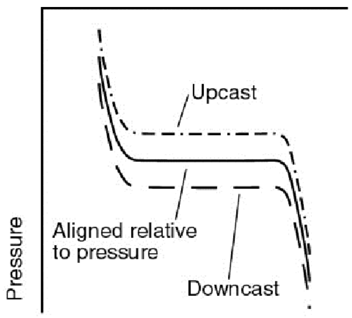
Align CTD

Note:
Align CTD cannot be run on files that have been averaged into pressure or depth bins in Bin Average. If alignment is necessary, run Align CTD before running Bin Average.

Align CTD aligns parameter data in time, relative to pressure. This ensures that calculations of salinity, dissolved oxygen concentration, and other parameters are made using measurements from the same parcel of water. Typically, Align CTD is used to align temperature, conductivity, and oxygen measurements relative to pressure.

There are three principal causes of misalignment of CTD measurements:

- physical misalignment of the sensors in depth
- inherent time delay (time constants) of the sensor responses
- water transit time delay in the pumped plumbing line - the time it takes the parcel of water to go through the plumbing to each sensor (or, for free-flushing sensors, the corresponding flushing delay, which depends on profiling speed)

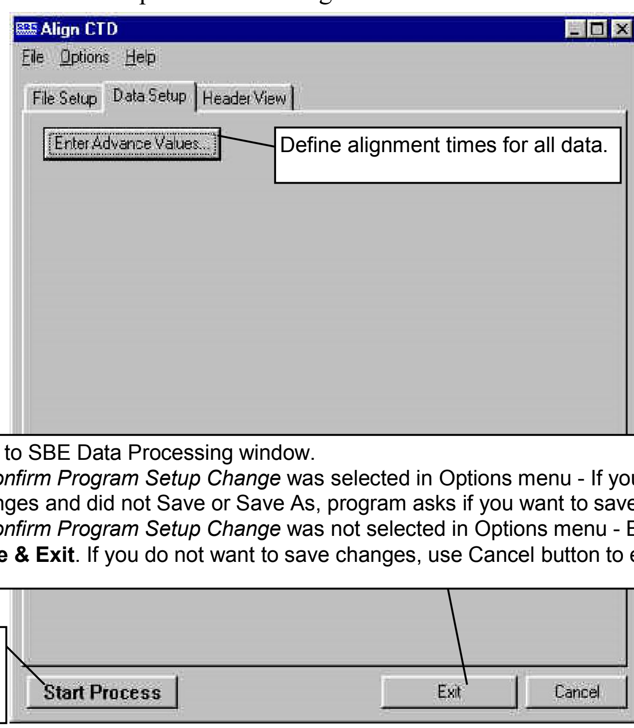


Upcast and Downcast mismatch with Respect to Pressure

When measurements are properly aligned, salinity spiking (and density) errors are minimized, and oxygen data corresponds to the proper pressure (e.g., temperature vs. oxygen plots agree between down and up profiles).

Note:
The File Setup tab and Header View tab are similar for all modules; see *Section 2: Installation and Use*.

The Data Setup tab in the dialog box looks like this:

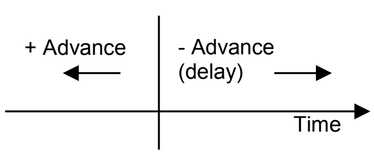


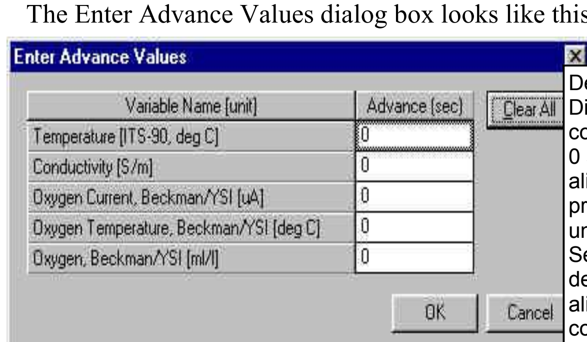
Define alignment times for all data.

Return to SBE Data Processing window.
- If *Confirm Program Setup Change* was selected in Options menu - If you made changes and did not Save or Save As, program asks if you want to save changes.
- If *Confirm Program Setup Change* was not selected in Options menu - Button says **Save & Exit**. If you do not want to save changes, use Cancel button to exit.

Begin processing data. Status field on File Setup tab shows *Processing complete* when done.

The Enter Advance Values dialog box looks like this:





Define alignment times. Diagram shows sign convention for Advance. If 0 seconds is entered, alignment relative to pressure (and time) remains unchanged for that variable. See discussion below to determine appropriate alignment times for conductivity, temperature, and oxygen.

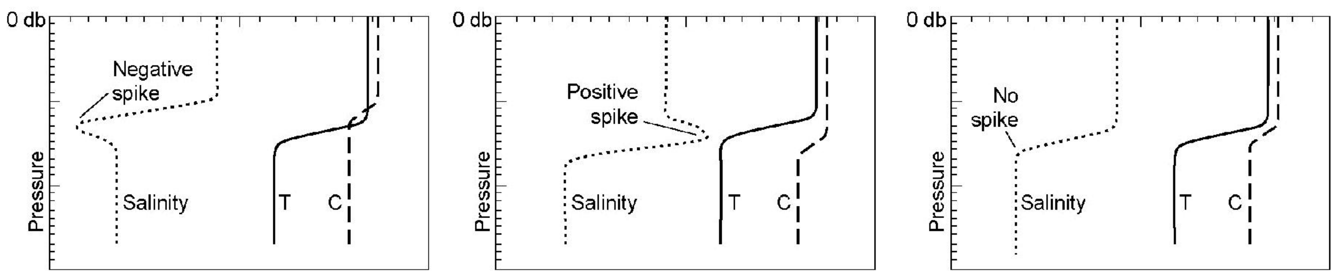
Align CTD: Conductivity and Temperature

Temperature and conductivity are often misaligned with respect to pressure. Shifting temperature and conductivity relative to pressure can compensate. As shown in the figures, indications of misalignment include:

- Depth mismatch between downcast and upcast data
- Spikes in the calculated salinity (which is dependent on temperature, conductivity, and pressure) - caused by misalignment of temperature and conductivity *with each other*

The best diagnostic of proper alignment is the elimination of salinity spikes that coincide with very sharp temperature steps. To determine the best alignment, plot 10 meters of temperature and salinity data at a depth that contains a very sharp temperature step. For the downcast, when temperature and salinity decrease with increasing pressure:

- A negative salinity spike at the conductivity step means that conductivity leads temperature (conductivity sensor *sees* step before temperature sensor does). Advance conductivity *relative to temperature* a **negative** number of seconds.
- Conversely, if the salinity spike is positive, advance conductivity *relative to temperature* a **positive** number of seconds.



Downcast, Conductivity leads Temperature

Downcast, Conductivity lags Temperature

Downcast, C and T Aligned

The best alignment of conductivity with respect to temperature is obtained when the salinity spikes are minimized. Some experimentation with different advances is required to find the best alignment.

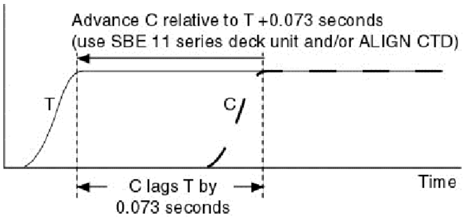
Typical Temperature Alignment

The SBE 19, 19*plus*, and 19*plus* V2 use a temperature sensor with a relatively slow time response, while the SBE 9*plus*, 25, 25*plus*, and 49 use a temperature sensor with a faster time response. Typical advances are:

Instrument	Advance of Temperature Relative to Pressure (seconds)
9*plus*	0
19, 19*plus*, or 19*plus* V2	+ 0.5
25 or 25*plus*	0
49 *	+ 0.0625

*The SBE 49 can be programmed to advance temperature relative to pressure in real-time, eliminating the need to run Align CTD. See the SBE 49 manual for details.

Note:
All SBE 11 series deck units can advance **primary** conductivity, which *may* eliminate the need to use Align CTD for conductivity. The SBE 11*plus* does not advance secondary conductivity. The SBE 11*plus* V2 can advance secondary conductivity and all voltage channels; the advance time is user-programmable.



Typical Conductivity Alignment

- SBE 9*plus* - For an SBE 9*plus* with TC-ducted temperature and conductivity sensors and a 3000-rpm pump, the typical lag of conductivity relative to temperature is 0.073 seconds. The Deck Unit can be programmed to advance conductivity relative to pressure, eliminating the need to run Align CTD.
  Following is an example of determining the value to enter in Align CTD:
  Example: The SBE 11*plus* is factory-set to advance the primary conductivity +1.75 scans (at 24 Hz, this is 1.75 / 24 = 0.073 seconds). Advance conductivity relative to temperature in Align CTD:
  0.073 - 1.75/24 = 0.0 seconds (enter 0 seconds for conductivity).

- SBE 19*plus* or 19*plus* V2 – For an SBE 19*plus* or 19*plus* V2 with a standard 2000-rpm pump, do not advance conductivity.

- SBE 19 (not *plus*) – For an unpumped SBE 19, the conductivity measurement may lead or lag that of temperature, because the flushing rate of the conductivity cell depends on drop speed. If the SBE 19 is lowered very slowly (< 20 cm/second, typically from a fixed platform or ice), conductivity lags temperature. If the SBE 19 is lowered fast, conductivity leads temperature. Typical advances of conductivity *relative to temperature* range from 0 seconds at a lowering rate of 0.75 meters/second to -0.6 seconds for 2 meters/second (if temperature was advanced +0.5 seconds, these correspond to conductivity advances of +0.5 seconds and -0.1 seconds respectively).

- SBE 25 or 25*plus* - For an SBE 25 or 25*plus* with a standard 2000-rpm pump, a typical advance of conductivity *relative to temperature* is +0.1 seconds.

- SBE 49 – For a typical SBE 49 with TC duct and 3000 rpm pump, do not advance conductivity.

If temperature is advanced relative to pressure and you do not want to change the relative timing of temperature and conductivity, you must add the same advance to conductivity.

Example (typical of an unpumped SBE 19):

Advance temperature relative to pressure +0.5 seconds to compensate for slow response time of sensor.

- If the CTD is lowered at 0.75 m/s, advance conductivity *relative to temperature* 0 seconds. Calculate advance of conductivity *relative to pressure* to enter in Align CTD: +0.5 + 0 = +0.5 seconds
- If the CTD is lowered at 2 m/s, advance conductivity *relative to temperature* -0.6 seconds. Calculate advance of conductivity *relative to pressure* to enter in Align CTD: +0.5 + (-0.6) = -0.1 seconds

Align CTD: Oxygen

Oxygen data is also systematically delayed with respect to pressure. The two primary causes are the long time constant of the oxygen sensor (for the SBE 43, ranging from 2 seconds at 25 ºC to approximately 5 seconds at 0 ºC) and an additional delay from the transit time of water in the pumped plumbing line. As with temperature and conductivity, you can compensate for this delay by shifting oxygen data relative to pressure. Typical advances for the SBE 43, 13, or 23 are:

Instrument	Advance of Oxygen Relative to Pressure (seconds)
9*plus*	+2 to +5
19*plus* or 19*plus* V2	+3 to +7
19 (not *plus*)	+3 to +7 (pumped), +1 to +5 (unpumped)
25 or 25*plus*	+3 to +7

Align CTD adds the following to the data file header:

Label	Description
Alignctd_date	Date and time that module was run.
Alignctd_in	Input .cnv converted data file.
Alignctd_adv	Variables aligned and their respective alignment times.

Bin Average

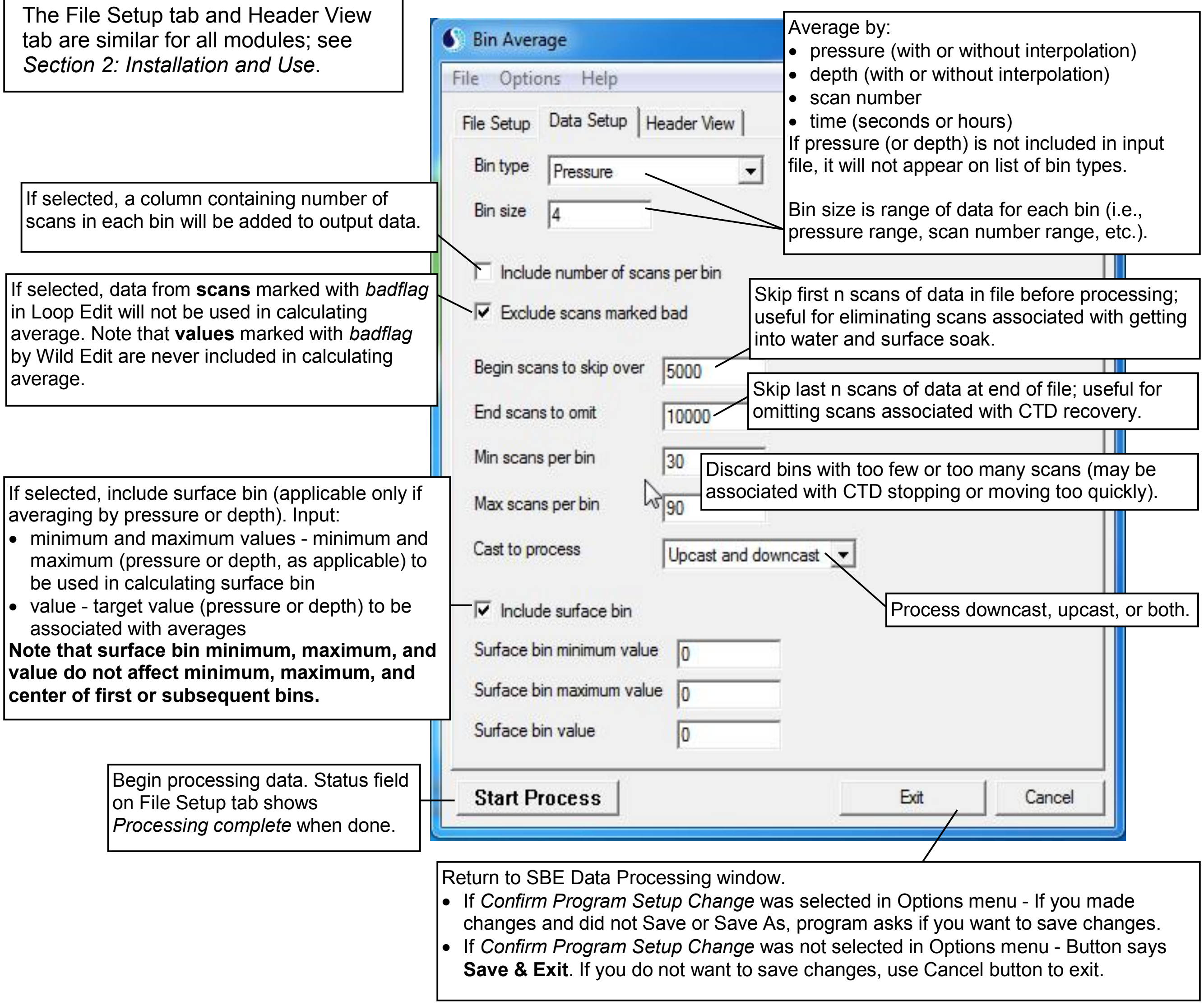
Note:
Align CTD, which aligns parameter data in time, relative to pressure, cannot be run on files that have been averaged into pressure or depth bins in Bin Average. If alignment is necessary, run Align CTD before running Bin Average.

Bin Average averages data, using averaging intervals based on:

- pressure range,
- depth range,
- scan number range, or
- time range

Note:
The File Setup tab and Header View tab are similar for all modules; see *Section 2: Installation and Use*.

The Data Setup tab in the dialog box looks like this:

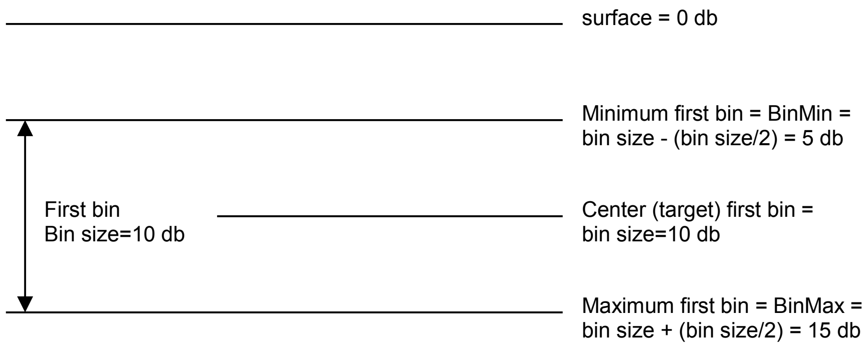


Bin Average: Formulas

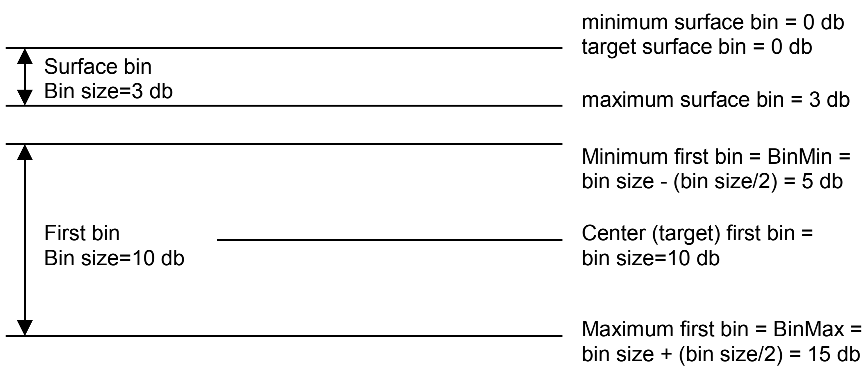
Note:
If *Exclude scans marked bad* is selected in the dialog box, data from **scans** marked with *badflag* in Loop Edit are not used in calculating average. **Values** marked with *badflag* by Wild Edit are never included in calculating the average. If the number of points included in the average is 0 (all data and/or scans in the bin are marked with *badflag*), the average value is set to *badflag.*

The center value of the first (not surface) bin is set equal to the bin size. The surface bin, if included, cannot overlap the first bin.

Example (pressure bin, surface bin not included):
Bin size is 10 db. The first bin is defined as follows:



Example (pressure bin, surface bin included):
Bin size is 10 db. Surface bin is included, and surface bin parameters are 0 db minimum, 3 db maximum, and 0 db value. The bins are defined as follows:



Note that for this example, the surface bin could have a maximum of up to 5 db (the minimum value for the first bin).

The algorithms used for each type of averaging follow.

Pressure Bins (no interpolation)
For each bin:
BinMin = center value - (bin size / 2)
BinMax = center value + (bin size / 2)

1. Add together valid data for scans with BinMin < pressure ≤ BinMax.
2. Divide sum by the number of valid data points to obtain average, and write average to output file.
3. Repeat Steps 1 through 2 for each variable.
4. For next bin, compute center value and repeat Steps 1 through 3.

Pressure Bins (with interpolation)

For each bin:
BinMin = center value - (bin size / 2)
BinMax = center value + (bin size / 2)

1. Add together valid data for scans with BinMin < pressure $\leq$ BinMax.
2. Divide sum by number of valid data points to obtain average.
3. Interpolate as follows, and write interpolated value to output file:
   P_p =average pressure of previous bin
   X_p =average value of variable in previous bin
   P_c =average pressure of current bin
   X_c =average value of variable in current bin
   P_i = center value for pressure in current bin
   X_i =interpolated value of variable (value at center pressure P_i)
   $= ((X_c - X_p) * (P_i - P_p) / (P_c - P_p)) + X_p$
4. Repeat Steps 1 through 3 for each variable.
5. Compute center value and Repeat Steps 1 through 4 for next bin.

Values for first bin are interpolated *after* averages for second bin are calculated; values from *next* (second) bin instead of *previous* bin are used in equations.

Depth Bins (with or without interpolation)

Depth bin processing is similar to processing pressure bins, but bin size and center values are based on depth.

Scan Number Bins

Scan number bin processing is similar to processing pressure bins without interpolation. If *exclude scans marked bad* is selected, Bin Average averages *bin size* good scans (not marked with *badflag* in Loop Edit).

Example: Bin size is 100. First bin should include scans 50-149. However, scans 93, 94, and 126 are marked with *badflag* in Loop Edit, and user selected *exclude scans marked bad*. To include 100 valid scans in average, Bin Average includes scans 50 - 152 in first bin.

Time Bins

Time bin processing is similar to processing pressure bins without interpolation. Bin Average determines the number of scans to include based on the input bin size and the data sampling interval:

Number of scans = bin size [seconds] / interval ***or***
Number of scans = (bin size [hours] x 3600 seconds/hour) / interval

Bin Average has the following /x parameter when run from the Command Line Options dialog box, from the command line, or with batch file processing:

/x Parameter	Description
/xbinavg:cN	N = center value for first bin.

See *Appendix I: Command Line Options, Command Line Operation, and Batch File Processing* for details on using parameters.

Bin Average adds the following to the data file header:

Label	Description
Binavg_date	Date and time that module was run.
Binavg_in	Input .cnv converted data file.
Binavg_bintype	Bin type (pressure, depth, scan time in seconds or hours).
Binavg_binsize	Bin size.
Binavg_excl_ bad_scans	If yes, values from scans marked with *badflag* in Loop Edit are not included in average.
Binavg_skipover	Number of scans skipped at beginning of file.
Binavg_omit	Number of scans skipped at end of file.
Binavg_min_ scans_bin	Minimum number of scans/bin; bins with fewer scans are discarded.
Binavg_max_ scans_bin_	Maximum number of scans/bin; bins with more scans are discarded.
Binavg_surface_ bin	Surface bin included? Minimum and maximum values for surface bin.

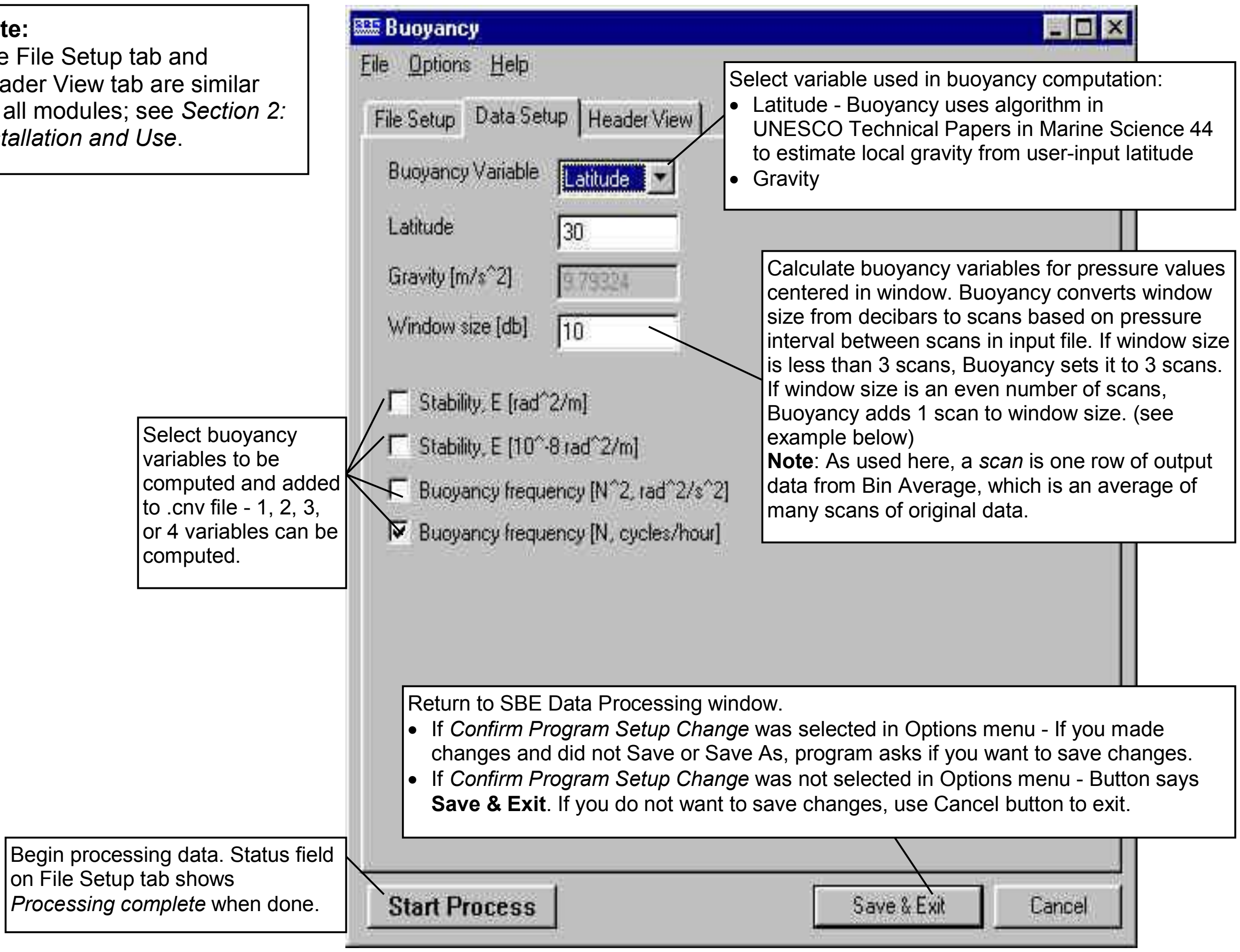
Buoyancy

Note:
The input .cnv file for Buoyancy must have been processed with Bin Average on pressure bins (with or without interpolation) and must contain pressure, temperature, and either salinity or conductivity.

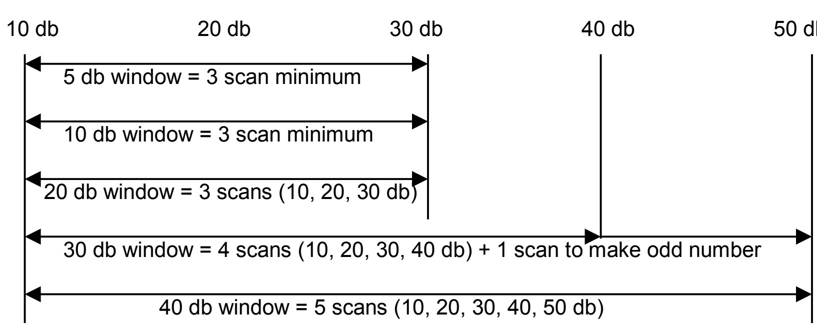
Buoyancy calculates buoyancy (Brunt-Väisälä) frequency (N) and stability (E) using the Fofonoff adiabatic leveling method (Bray N. A. and N. P. Fofonoff (1981) Available potential energy for MODE eddies. *Journal of Physical Oceanography*, 11, 30-46.).

The Data Setup tab in the dialog box looks like this:

Note:
The File Setup tab and Header View tab are similar for all modules; see *Section 2: Installation and Use*.



Example: For an interval of 10 db between scans, buoyancy window sizes of 5, 10, or 20 db result in a window size of 3 scans. Window sizes of 30 or 40 db result in a window size of 5 scans.



Buoyancy: Formulas

The relationship between frequency N and stability E is:

$N^2 = gE \quad [rad^2/s^2]$

where g = gravity [m / s^2]

The algorithm used to compute N^2 for the pressure value centered in the buoyancy window is:

1. Compute averages:
   p_bar = average pressure in the buoyancy window [decibars]
   t_bar = average temperature in the buoyancy window [deg C]
   s_bar = average salinity in the buoyancy window [PSU]
   rho_bar = density (s_bar, t_bar, p_bar) [Kg / m^3]

2. Compute the vertical gradient:
   theta = potential temperature (s, t, p, p_bar)
   v = 1 / density(s, theta, p_bar)
   where s, t, and p are the averaged values for salinity, temperature, and pressure calculated in Bin Average

   Use a least squares fit to compute the linear gradient dv/dp in the buoyancy window.

3. Compute N^2, N, E, and 10^{-8}E:

$$N^2 = -1.0e^{-4}\, rho\_bar^2\, g^2\, \frac{\delta v}{\delta p} \qquad [rad^2/s^2]$$

$$N = \frac{3600}{2\Pi}\sqrt{N^2} \qquad [cycles/hour]$$

$$E = \frac{N^2}{g} \qquad [rad^2/m]$$

$$E = 10^8\, \frac{N^2}{g} \qquad [10^{-8}\, rad^2/m]$$

Buoyancy adds the following to the data file header:

Label	Description
Buoyancy_date	Date and time that module was run.
Buoyancy_in	Input .cnv converted data file.
Buoyancy_vars	Gravity value (input value or value based on input latitude) and buoyancy window size (adjusted to provide a minimum of three scans and an odd number of scans).

Cell Thermal Mass

> **Note:**
> Cell thermal mass corrections should **not be applied to freshwater data**. It can give bad results, due to the way the derivative dC/dT is calculated in regions where conductivity changes are very small.

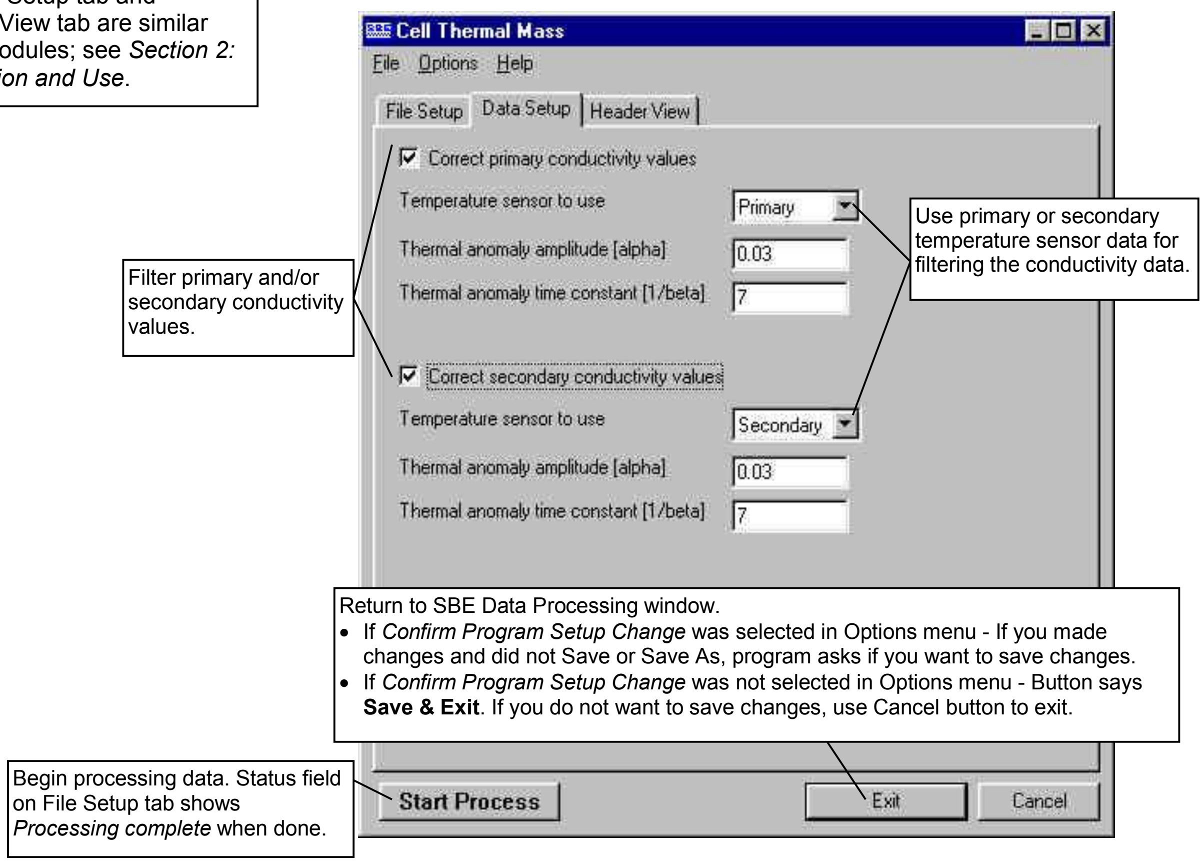
Cell Thermal Mass uses a recursive filter to remove conductivity cell thermal mass effects from the measured conductivity. Typical values for alpha and 1/beta are:

Instrument	alpha	1/beta
SBE 9*plus* with TC duct and 3000 rpm pump	0.03	7.0
SBE 19*plus* or 19*plus* V2 with TC duct and 2000 rpm pump	0.04	8.0
SBE 19 (not *plus*) with TC duct and 2000 rpm pump	0.04	8.0
SBE 19 (not *plus*) with no pump, moving at 1 m/sec	0.042	10.0
SBE 25 or 25*plus* with TC duct and 2000 rpm pump	0.04	8.0
SBE 49 with TC duct and 3000 rpm pump *	0.03	7.0

*The SBE 49 can be programmed to correct for conductivity cell thermal mass effects in real-time, eliminating the need to run Cell Thermal Mass. See the SBE 49 manual for details.

> **Note:**
> The File Setup tab and Header View tab are similar for all modules; see *Section 2: Installation and Use*.

The Data Setup tab in the dialog box looks like this:



Cell Thermal Mass: Formulas

The algorithm used is:

a = 2 * alpha / (sample interval * beta + 2)
b = 1 - (2 * a / alpha)
dc/dT = 0.1 * (1 + 0.006 * [temperature - 20])
dT = temperature - previous temperature
ctm [S/m] = -1.0 * b * previous ctm + a * (dc/dT) * dT

where
sample interval is measured in seconds and temperature in °C
ctm is calculated in S/m

If the input file contains conductivity in units other than S/m, Cell Thermal Mass applies the following scale factors to the calculated ctm:
ctm [mS/cm] = ctm [S/m] * 10.0
ctm [µS/cm] = ctm [S/m] * 10000.0

corrected conductivity = c + ctm

To determine the values for alpha and beta, see:
Lueck, R.G., 1990: Thermal Inertia of Conductivity Cells: Theory., American Meteorological Society Oct 1990, 741-755.

Cell Thermal Mass adds the following to the data file header:

Label	Description
Celltm_date	Date and time that module was run.
Celltm_in	Input .cnv converted data file.
Celltm_alpha	Value used for alpha.
Celltm_tau	Value used for 1/beta.
Celltm_temp_sensor _use_for_cond	Temperature sensor for primary conductivity filter, temperature sensor for secondary conductivity filter.

Derive (EOS-80; Practical Salinity)

Notes:
- Derive's File Setup tab **requires** selection of an input data file and instrument configuration (.con or .xmlcon) file. **SBE 37** stores calibration coefficients internally, and does not have a .con or .xmlcon file provided by Sea-Bird.
  - If you used SeatermV2 version 1.1 or later to upload SBE 37 data, the software created a .xmlcon file when it created the .hex file.
  - If you used an earlier version of SeatermV2 or any version of Seaterm to upload SBE 37 data, use a .con or .xmlcon file from **any** other Sea-Bird instrument; the contents will not affect the results. If you do not have a .con or .xmlcon file for another instrument, create one in SBE Data Processing's Configure menu (select **any** instrument in the Configure menu, then click Save As in the Configuration dialog box).
- Algorithms used for calculation of derived parameters in Data Conversion, Derive, Sea Plot, SeaCalc III [EOS-80 (Practical Salinity) tab], and Seasave are identical, except as noted in *Appendix V: Derived Parameter Formulas (EOS-80; Practical Salinity)*, and are based on EOS-80 equations.
- Derive is not compatible with a .cnv file from an **SBE 39**, **39-IM**, **39*plus***, **39*plus*-IM, or 48**.
- For an SBE 21 or 45 with a remote temperature sensor, Seasave, Data Conversion, Derive, and Derive TEOS-10 all use the remote temperature data when calculating density and sound velocity.

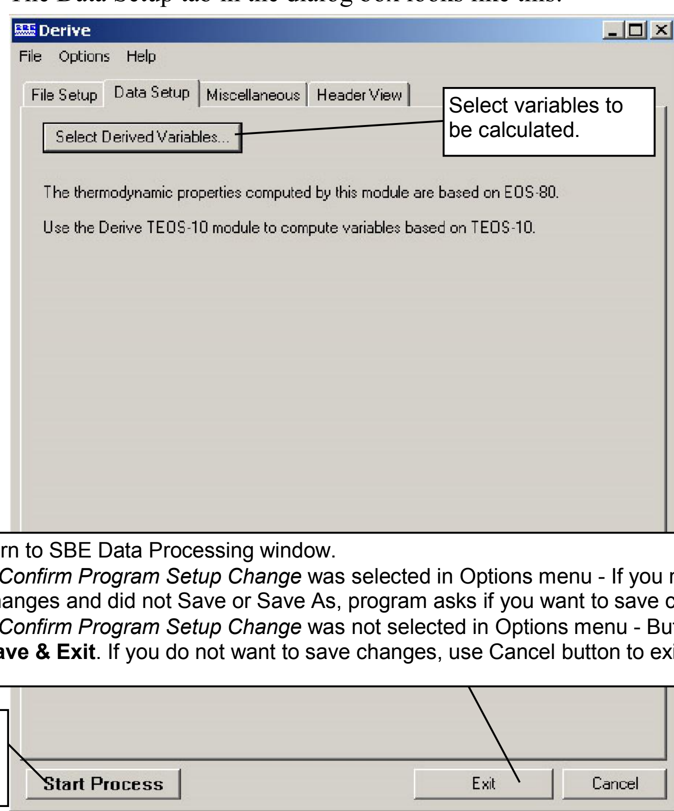
Derive uses pressure, temperature, and conductivity from the input .cnv file to compute the following oceanographic parameters:
- density (density, sigma-theta, sigma-1, sigma-2, sigma-4, sigma-t)
- thermosteric anomaly
- specific volume
- specific volume anomaly
- geopotential anomaly
- dynamic meters
- depth (salt water, fresh water)
- salinity
- sound velocity (Chen-Millero, DelGrosso, Wilson)
- average sound velocity
- potential temperature (reference pressure = 0.0 decibars)
- potential temperature anomaly
- specific conductivity
- derivative variables (descent rate and acceleration) - if input file has not been averaged into pressure or depth bins
- oxygen (if input file contains pressure, temperature, and either conductivity or salinity, and has not been averaged into pressure or depth bins) - also requires oxygen current and oxygen temperature (SBE 13 or 23) or oxygen signal (SBE 43)
- corrected irradiance (CPAR)

See *Appendix V: Derived Parameter Formulas* for the formulas used to calculate these parameters.

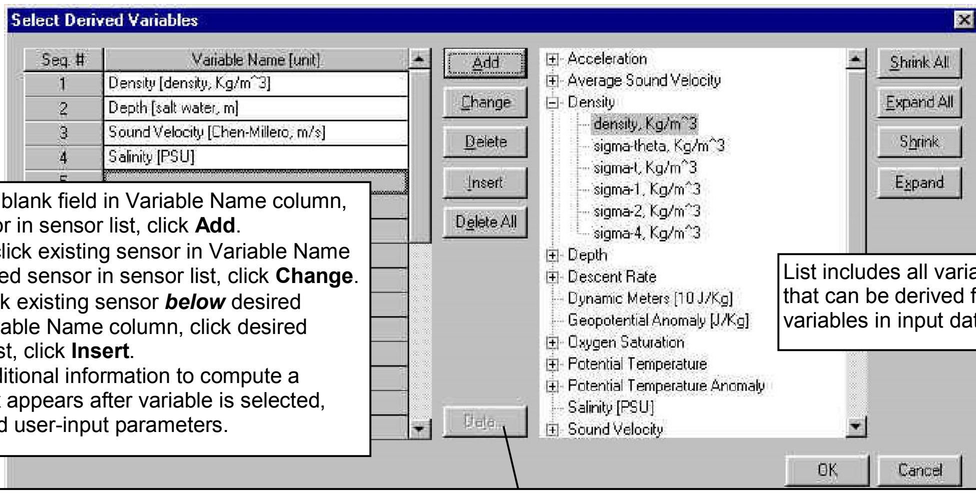
See **Derive TEOS-10** after this module to calculate TEOS-10 (Absolute Salinity) parameters.

Note:
The File Setup tab and Header View tab are similar for all modules; see *Section 2: Installation and Use*.

The Data Setup tab in the dialog box looks like this:



The Select Derived Variables dialog box looks like this:



- Add variable: click blank field in Variable Name column, click desired sensor in sensor list, click **Add**.
- Change variable: click existing sensor in Variable Name column, click desired sensor in sensor list, click **Change**.
- Insert variable: click existing sensor ***below*** desired sequence # in Variable Name column, click desired sensor in sensor list, click **Insert**.

If Derive requires additional information to compute a variable, a dialog box appears after variable is selected, with fields for required user-input parameters.

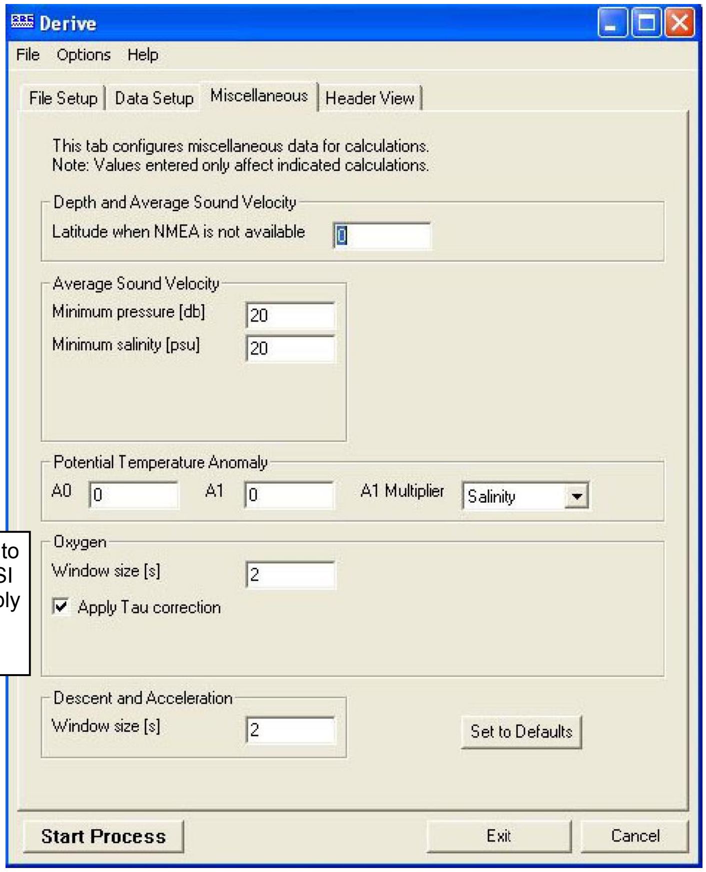
List includes all variables that can be derived from variables in input data file.

Click **Data** to view/modify user-input parameters for selected variable (if applicable). Some variables *share* a user-input parameter, so changing a parameter for one variable automatically changes it for the other:
- Depth and average sound velocity use same latitude (if NMEA data not available).
- Descent rate and acceleration use same time window size.
- All SBE 13, 23, and 43 oxygen sensors use same time window size, Tau correction, and (SBE 43 only) hysteresis correction.

Note: An alternate method of entering these parameters is on Miscellaneous tab in Derive dialog box.

The Miscellaneous tab in the Derive dialog box looks like this:

Note:
Values for these parameters can be changed on the Miscellaneous tab or by double clicking on the output variable in the Select Derived Variables dialog box (above); changes made in one location are automatically made in the other location.



Oxygen selections apply to SBE 43 and Beckman/YSI sensors. They do not apply to SBE 63 or Aanderaa Oxygen Optode.

The Miscellaneous tab defines parameters required for output of specific variables (depth, average sound velocity, potential temperature anomaly, oxygen, descent rate, and acceleration). Entries on this tab are used only if you are calculating and outputting the associated variable to the .cnv file. For example, if you do not select Oxygen in the Select Derived Variables dialog box, Derive ignores the value entered for Oxygen window size and the enabling of the Tau correction on the Miscellaneous tab.

In Derive, derivative variables (oxygen, descent rate, and acceleration) are computed by looking at data centered around the current data point with a time span equal to the user-input time window size and using a linear regression to determine the slope. This differs from how the calculation is done in Seasave and Data Conversion, which compute the derivative looking backward in time, since they share common code and Seasave cannot use future values while acquiring data in real-time.

Derive has the following /x parameter when run from the Command Line Options dialog box, from the command line, or with batch file processing:

/x Parameter	Description
/xderive:pump	For SBE 911*plus*, do not output scans if pump status = off.

See *Appendix I: Command Line Options, Command Line Operation, and Batch File Processing* for details on using parameters.

Derive adds the following to the data file header:

Label	Description
Derive_date	Date and time that module was run. Also shows how many columns of data (how many variables) were derived.
Derive_in	Input .cnv converted data file and .con or .xmlcon configuration file.
Derive_time_window_docdt	Window size for oxygen derivative calculation (seconds).
Derive_time_window_dzdt	Window size for descent rate and acceleration calculation (seconds).
Derive_ox_tau_ correction	Whether tau correction was performed on oxygen data.

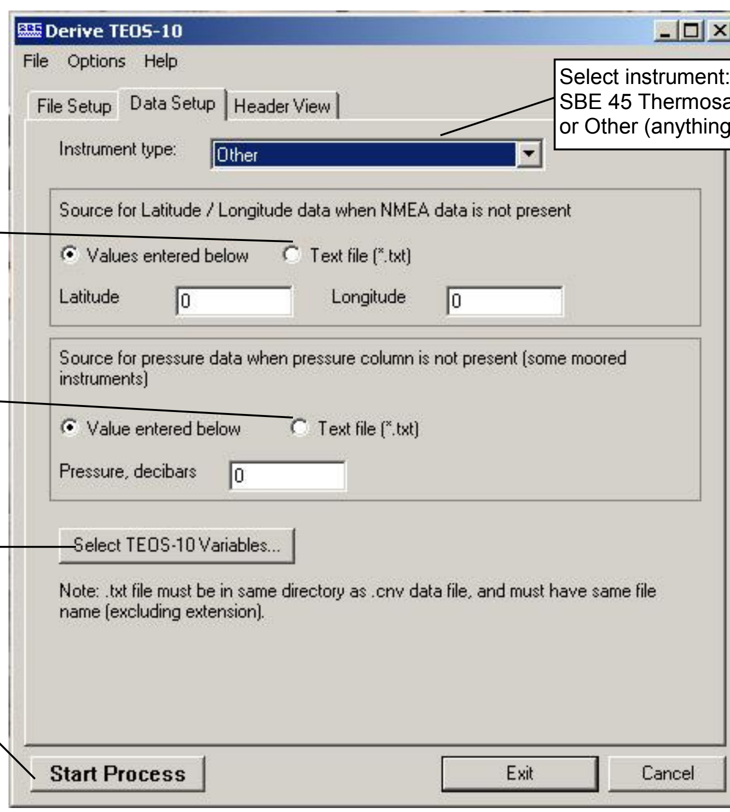
Derive TEOS-10

Notes:
- Algorithms used in Derive TEOS-10 are based on the TEOS-10 website: www.TEOS-10.org.
- Derive TEOS-10 is not compatible with a .cnv file from an **SBE 39, 39-IM, 39*plus*, 39*plus*-IM, or 48**.
- For an SBE 21 or 45 with a remote temperature sensor, Seasave, Data Conversion, Derive, and Derive TEOS-10 all use the remote temperature data when calculating density and sound velocity.

Derive TEOS-10 uses temperature, conductivity **or** salinity (Practical, EOS-80), pressure, latitude, and longitude to compute the following thermodynamic parameters using TEOS-10 equations:
- Absolute Salinity
- Absolute Salinity Anomaly
- adiabatic lapse rate
- Conservative Temperature
- Conservative Temperature freezing
- density
- dynamic enthalpy
- enthalpy
- entropy
- gravity
- internal energy
- isentropic compressibility
- latent head of evaporation
- latent heat of melting
- potential temperature
- Preformed Salinity
- Reference Salinity
- saline contraction coefficient
- sound speed
- specific volume
- specific volume anomaly
- temperature freezing
- thermal expansion coefficient

Notes:
- The File Setup tab and Header View tab are similar for all modules; see *Section 2: Installation and Use*.
- (if used) Values for longitude, latitude, and pressure in .txt files are not limited to the number of digits shown in the examples.

The Data Setup tab in the dialog box looks like this:



Select instrument: SBE 21 or SBE 45 Thermosalinograph, or Other (anything else).

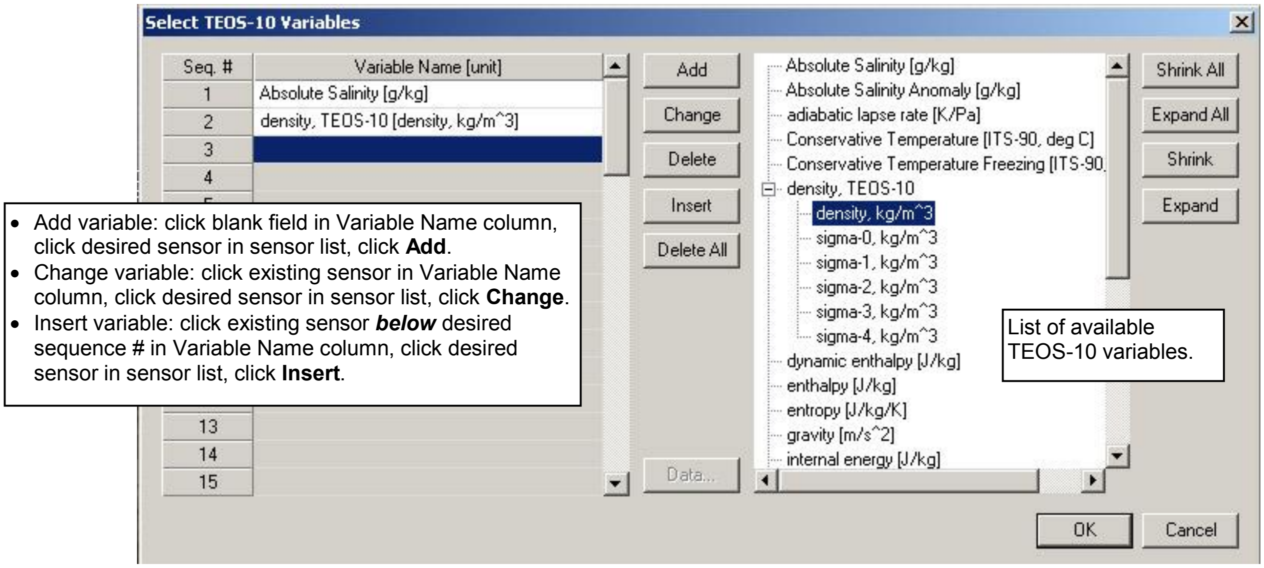
.txt file format is:
//comment line starts with two //
Longitude: xxx.xx
Latitude: xxx.xx

.txt file format is:
//comment line starts with two //
Pressure: xxxxx.xx

Select variables to be calculated (see below).

Begin processing data. Status field on File Setup tab shows *Processing complete* when done.

The Select TEOS-10 Variables dialog box looks like this:



Derive TEOS-10 adds the following to the data file header:

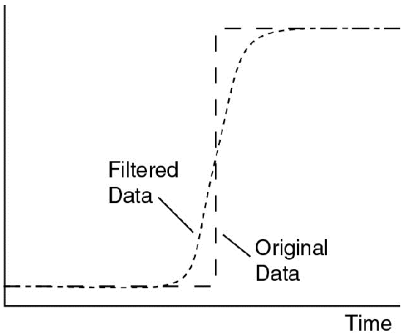
Label	Description
DeriveTEOS_10_date	Date and time that module was run. Also shows how many columns of data (how many variables) were derived.
DeriveTEOS_10_in	Input .cnv converted data file
DeriveTEOS_10_latitude_source	Source of latitude data.
DeriveTEOS_10_longitude_source	Source of longitude data.
Using the GSW Toolkit version xx.xx	Source and version of equations used in TEOS-10 calculations.

TEOS-10 Formulas

The following table references the C functions from www.TEOS-10.org that are implemented in Derive TEOS-10:

SBE Data Processing variable name (in Select TEOS-10 Variables dialog and in output .cnv file)	C function from www.TEOS-10.org code
Absolute Salinity	gsw_sa_from_sp
Absolute Salinity Anomaly	gsw_deltasa_from_sp
adiabatic lapse rate	gsw_adiabatic_lapse_rate_from_ct
Conservative Temperature	gsw_ct_from_t
Conservative Temperature freezing	gsw_ct_freezing
density, TEOS-10	gsw_rho (use gsw_rho with reference pressure for the sigmas)
dynamic enthalpy	gsw_dynamic_enthalpy
enthalpy	gsw_enthalpy
entropy	gsw_entropy_from_t
gravity	gsw_grav
internal energy	gsw_internal_energy
isentropic compressibility	gsw_kappa
latent heat of evaporation	gsw_latentheat_evap_ct
latent heat of melting	gsw_latentheat_melting
potential temperature	gsw_pt0_from_t
Preformed Salinity	gsw_sstar_from_sa
Reference Salinity	gsw_sr_from_sp
saline contraction coefficient	gsw_beta
sound speed	gsw_sound_speed
specific volume	gsw_specvol
specific volume anomaly	gsw_specvol_anom
temperature freezing	gsw_t_freezing
thermal expansion coefficient	gsw_alpha

Filter



Filter runs a low-pass filter on one or more columns of data. A low-pass filter smoothes high frequency (rapidly changing) data. To produce zero phase (no time shift), the filter is first run forward through the data and then run backward through the forward-filtered data. This removes any delays caused by the filter.

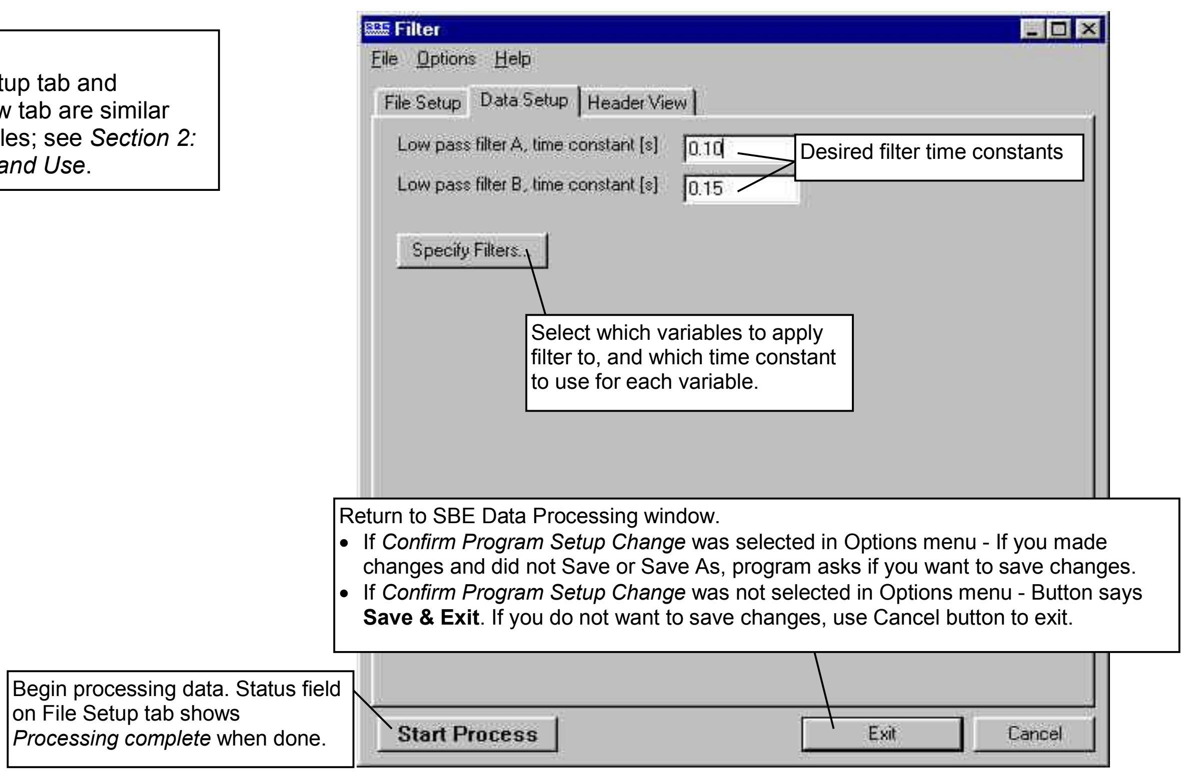
Pressure data is typically filtered with a time constant equal to four times the CTD scan rate. Conductivity and temperature are typically filtered for *some* CTDs. Two time constants can be specified, so different parameters can be filtered with different time constants in one run of Filter. Typical time constants are:

Instrument	Temperature (seconds)	Conductivity (seconds)	Pressure (seconds)
SBE 9*plus*	-	-	0.15
SBE 19*plus* or 19*plus* V2	0.5	0.5	1.0
SBE 19 (not *plus*) with or without TC duct and pump	0.5	0.5	2.0
SBE 25 or 25*plus*	0.1	0.1	0.5
SBE 49 with TC duct and 3000 rpm pump *	0.085	0.085	0.25

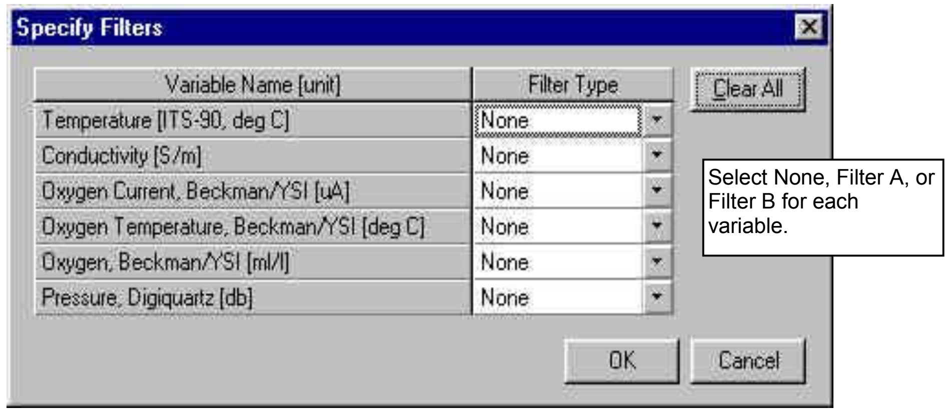
*The SBE 49 can be programmed to filter the data in real-time with a cosine window filter (see *WFilter*), eliminating the need to run Filter on temperature and conductivity data. See the SBE 49 manual for details.

Note:
The File Setup tab and Header View tab are similar for all modules; see *Section 2: Installation and Use*.

The Data Setup tab in the dialog box looks like this:



The Specify Filters dialog box looks like this:



Filter: Formulas

For a low-pass filter with time constant Γ:

$\Gamma = 1/\omega \qquad \omega = 2\pi f$
T = sample interval (seconds)
$S_0 = 1/\Gamma$

Laplace transform of the transfer function of a low-pass filter (single pole) with a time constant of Γ seconds is:

$$H(s) = \frac{1}{1 + (S/S_0)}$$

Using the bilinear transform:

$$S - f(z) \stackrel{\Delta}{=} \frac{2\,(1-z^{-1})}{T\,(1+z^{-1})} = \frac{2\,(z-1)}{T\,(z+1)}$$

$$H(z) = \frac{1}{1 + \dfrac{2\,(z-1)}{T\,(z+1)\,S_0}} = \frac{z^{-1} + 1}{1 + \dfrac{2}{TS_0}\left\{1 + \left(\dfrac{1 - 2/TS_0}{1 + 2/TS_0}\right) z^{-1}\right\}}$$

If: $$A = \frac{1}{1 + \dfrac{2}{TS_0}} \qquad B = \frac{1 - \dfrac{2}{TS_0}}{1 + \dfrac{2}{TS_0}}$$

Then: $$H(z) = \frac{Y(z)}{X(z)} = \frac{A\,(z^{-1}+1)}{(1 + Bz^{-1})}$$

Where z^{-1} is the unit delay (one scan behind).

y[N] = current output
y[N-1] = previous output
x[N] = input data (current scan)
x[N-1] = previous input data (from previous scan)

$Y(z)\,(1 + Bz^{-1}) = X(z)\,A\,(z^{-1} + 1)$
$y[N] + By[N-1] = Ax[N-1] + Ax[N]$
$y[N] = A(x[N] + x[N-1]) - By[N-1]$

Example: Time constant = 0.5 second, sample interval = 1/24 second

$$A = \frac{1}{(1 + 2 * 0.5 * 24)} = \frac{1}{(1 + 24)} = 0.04$$

$$B = (1 - 2 * 0.5 * 24)\ A = \frac{1 - 24}{1 + 24} = -0.92$$

Filter adds the following to the data file header:

Label	Description
Filter_date	Date and time that module was run.
Filter_in	Input .cnv converted data file.
Filter_low_pass_tc_A	Time constant for filter A.
Filter_low-Pass_tc_B	Time constant for filter B.
Filter_low_pass_A_vars	List of variables filtered with time constant A.
Filter_low_pass_B_vars	List of variables filtered with time constant B.

Loop Edit

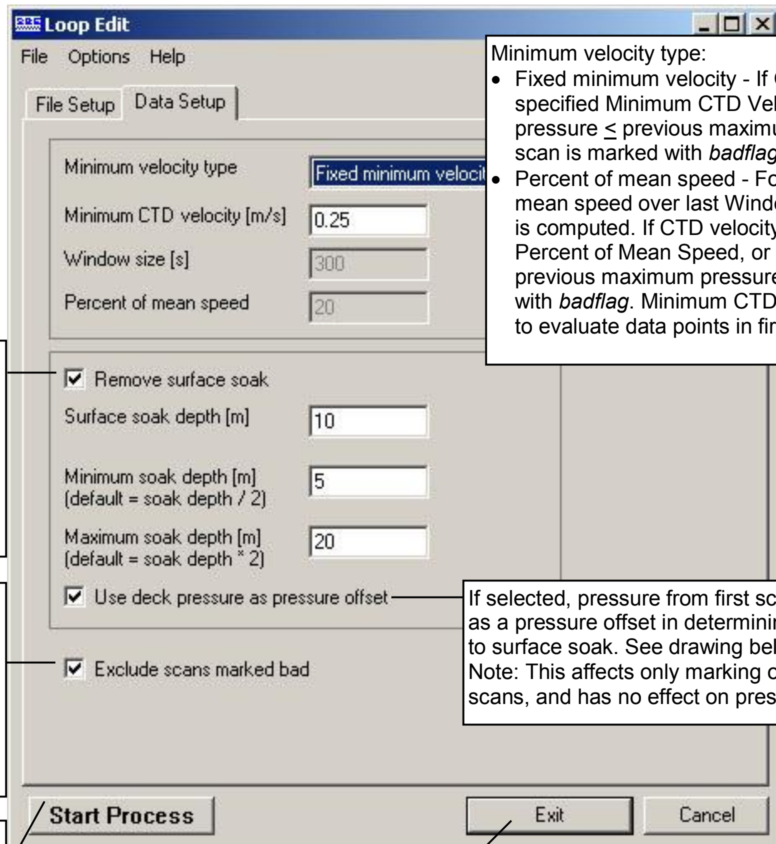
Loop Edit marks scans *bad* by setting the flag value associated with the scan to *badflag* in input .cnv files that have pressure slowdowns or reversals (typically caused by ship heave). Optionally, Loop Edit can also mark scans associated with an initial surface soak with *badflag*. The *badflag* value is documented in the input .cnv header.

Note:
Data Conversion calculates velocity with a 2-second window (e.g., 48 scans for an SBE 9*plus*), giving a much smoother measure of velocity.

Loop Edit operates on three successive scans to determine velocity. This is such a fine scale that noise in the pressure channel from counting jitter or other unknown sources can cause Loop Edit to mark scans with *badflag* in error. **Therefore, you must run Filter on the pressure data to reduce noise before you run Loop Edit**. See *Filter* for pressure filter recommendations for each instrument.

Note:
The File Setup tab and Header View tab are similar for all modules; see *Section 2: Installation and Use*.

The Data Setup tab in the dialog box looks like this:



Minimum velocity type:
- Fixed minimum velocity - If CTD velocity < specified Minimum CTD Velocity, or pressure ≤ previous maximum pressure, scan is marked with *badflag*.
- Percent of mean speed - For each scan, mean speed over last Window Size seconds is computed. If CTD velocity < specified Percent of Mean Speed, or pressure ≤ previous maximum pressure, scan is marked with *badflag*. Minimum CTD Velocity is used to evaluate data points in first time window.

If selected, scans related to surface soak are marked with *badflag*, based on Minimum soak depth and Maximum soak depth (note that Surface soak depth is not actually used in calculation of surface soak scans). See drawing below for details.

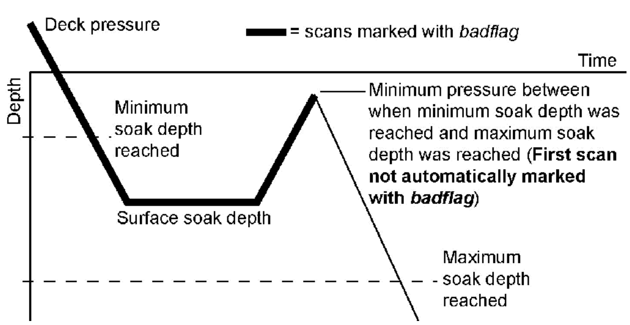
If selected, pressure from first scan in file is used as a pressure offset in determining scans related to surface soak. See drawing below for details. Note: This affects only marking of surface soak scans, and has no effect on pressure data in file.

- If selected, scans previously marked with *badflag* (for example, in a previous run of Loop Edit) will not be evaluated.
- If not selected, scans previously marked with *badflag* will be reevaluated, and scan's flag will be reset accordingly.

Begin processing data. Status field on File Setup tab shows *Processing complete* when done.

Return to SBE Data Processing window.
- If *Confirm Program Setup Change* was selected in Options menu - If you made changes and did not Save or Save As, program asks if you want to save changes.
- If *Confirm Program Setup Change* was not selected in Options menu - Button says **Save & Exit**. If you do not want to save changes, use Cancel button to exit.

Algorithm for removal of surface soak data



Loop Edit adds the following to the data file header:

Label	Description
Loopedit_date	Date and time that module was run.
Loopedit_in	Input .cnv converted data file.
Loopedit_minVelocity	If *Fixed Minimum Velocity* was selected - minimum CTD velocity for good scans; scans with velocity less than this are marked with *badflag*.
Loopedit_percentMeanSpeed	If *Percent of Mean Speed* was selected - minimum CTD velocity for first time window, window size, and percent of mean speed for good scans; scans that do not meet this criteria are marked with *badflag*.
Loopedit_surfaceSoak	If *Remove surface soak* was selected – minimum soak depth, maximum soak depth, and whether to use deck pressure as a pressure offset (1 = yes, 0 = no).
Loopedit_excl_bad_scans	If yes, do not evaluate scans marked with *badflag* in a previous run of Loop Edit.

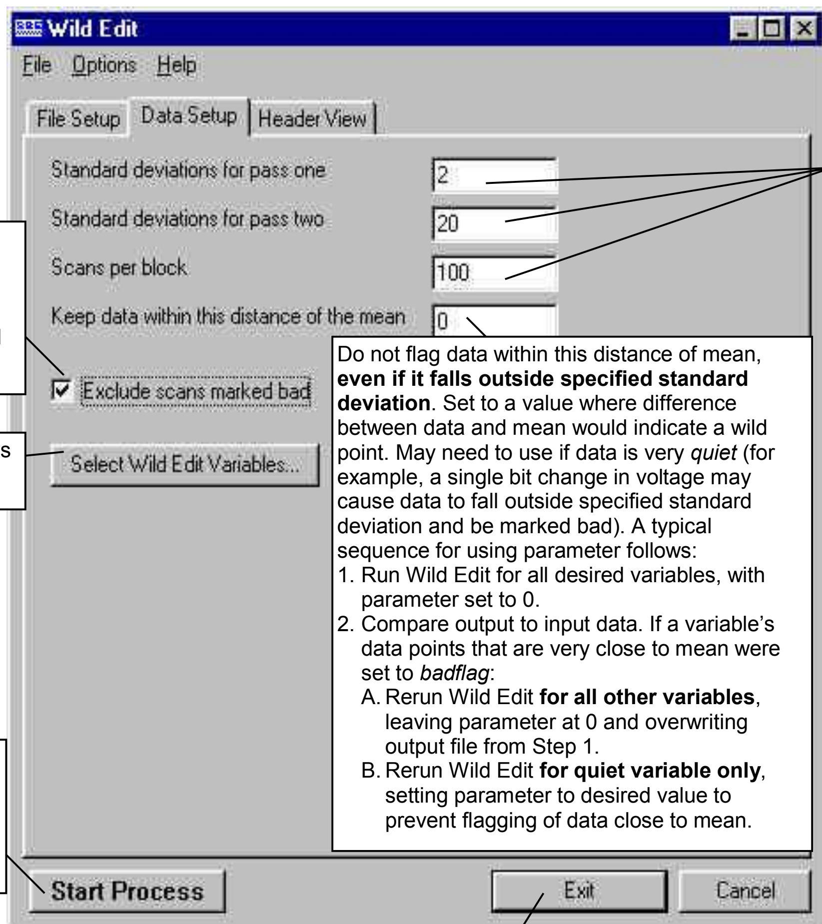
Wild Edit

Note:
Wild Edit marks **individual data** (for example, a conductivity value) with *badflag*, but does not mark the entire scan (which may include other data that is valid, such as temperature, pressure, etc.).

Note:
The File Setup tab and Header View tab are similar for all modules; see *Section 2: Installation and Use*.

Wild Edit marks wild points in the data by replacing the data value with *badflag*. The *badflag* value is documented in the input .cnv header.
Wild Edit's algorithm requires two passes through the data: the first pass obtains an accurate estimate of the data's true standard deviation, while the second pass replaces the appropriate data with *badflag*.

The Data Setup tab in the dialog box looks like this:



Wild Edit operates as follows:
1. Compute mean and standard deviation of data in block (specified by Scans per Block) for each selected variable. **Temporarily** flag values that differ from mean by more than standard deviations specified for pass 1.
2. Recompute mean and standard deviation, excluding temporarily flagged values. Mark values that differ from mean by more than standard deviations specified for pass 2 by replacing data value with *badflag*.
3. Repeat Steps 1 and 2 for next block of scans.
   - If last block of data in input file has less than specified number of scans, use data from previous block to fill in block.

If selected, data from scans marked with *badflag* in Loop Edit will not be used in calculating mean and standard deviation.

Select which variables to run Wild Edit on.

Do not flag data within this distance of mean, **even if it falls outside specified standard deviation**. Set to a value where difference between data and mean would indicate a wild point. May need to use if data is very *quiet* (for example, a single bit change in voltage may cause data to fall outside specified standard deviation and be marked bad). A typical sequence for using parameter follows:
1. Run Wild Edit for all desired variables, with parameter set to 0.
2. Compare output to input data. If a variable's data points that are very close to mean were set to *badflag*:
   A. Rerun Wild Edit **for all other variables**, leaving parameter at 0 and overwriting output file from Step 1.
   B. Rerun Wild Edit **for quiet variable only**, setting parameter to desired value to prevent flagging of data close to mean.

Begin processing data. Status field on File Setup tab shows *Processing complete* when done.

Return to SBE Data Processing window.
- If *Confirm Program Setup Change* was selected in Options menu - If you made changes and did not Save or Save As, program asks if you want to save changes.
- If *Confirm Program Setup Change* was not selected in Options menu - Button says **Save & Exit**. If you do not want to save changes, use Cancel button to exit.

If the data file is particularly corrupted, you may need to run Wild Edit more than once, with different block sizes and number of standard deviations.

If the input file has some variables with large values and some with relatively smaller values, it may be necessary to run Wild Edit more than once, varying the value for *Keep data within this distance of mean* so that it is meaningful for each variable. Better results may also be obtained by increasing *Scans per block* from 100 to around 500.

> *Example*
> Sensor A's range is approximately 1000 and Sensor B's range is approximately 10. Run Wild Edit on Sensor A, using *Keep data within this distance of mean* = 10. Then run Wild Edit on Sensor B, using *Keep data within this distance of mean* = 0.1

Wild Edit adds the following to the data file header:

Label	Description
Wildedit_date	Date and time that module was run.
Wildedit_in	Input .cnv converted data file.
Wildedit_pass1_nstd	Number of standard deviations for pass 1 test.
Wildedit_pass2_nstd	Number of standard deviations for pass 2 test.
Wildedit_pass2_mindelta	Keep data within this distance of mean.
Wildedit_npoint	Number of points to include in each test.
Wildedit_vars	List of the variables tested for wild points.
Wildedit_excl_bad_scans	If yes, values in scans marked with *badflag* (in Loop Edit) will not be used to determine standard deviation.

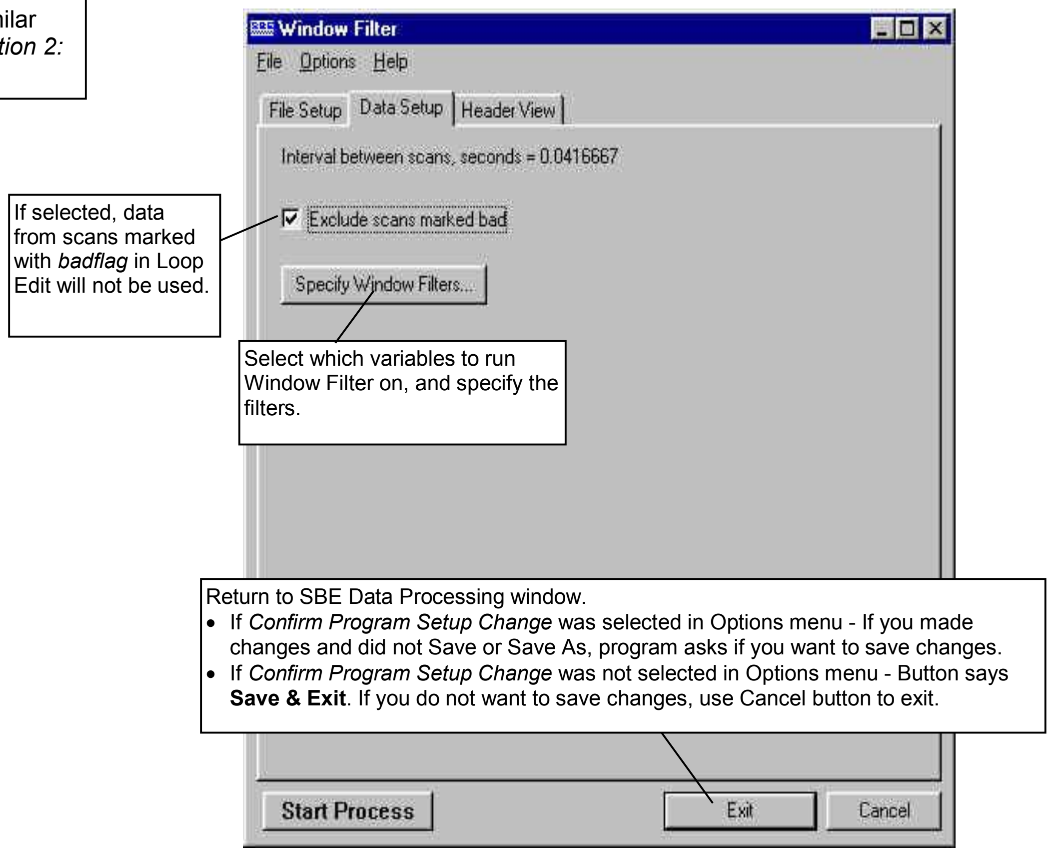
Window Filter

Window Filter provides four types of window filters and a median filter for data smoothing of .cnv files:

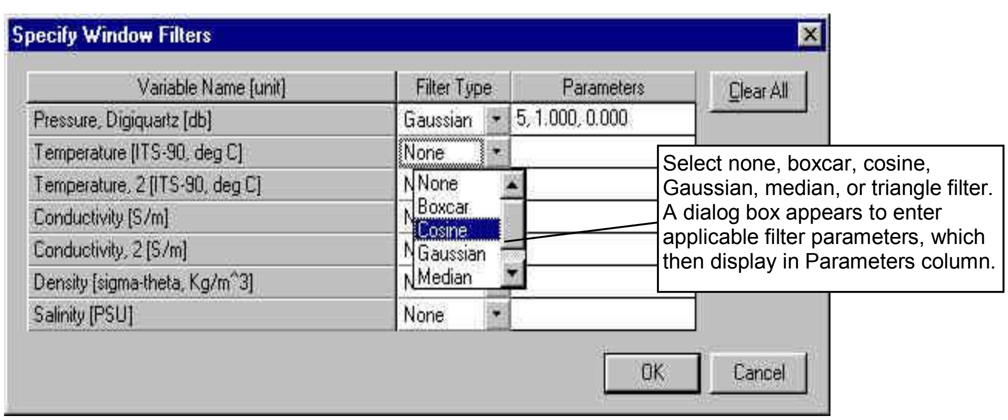
- Window filters calculate a weighted average of data values about a center point and replace the data value at the center point with this average.
- The median filter calculates a median for data values about a center point and replaces the data value at the center point with the median.

Note:
The File Setup tab and Header View tab are similar for all modules; see *Section 2: Installation and Use*.

The Data Setup tab in the dialog box looks like this:



The Specify Window Filters dialog box looks like this:



Window Filters: Descriptions and Formulas

Shape and length define filter windows:

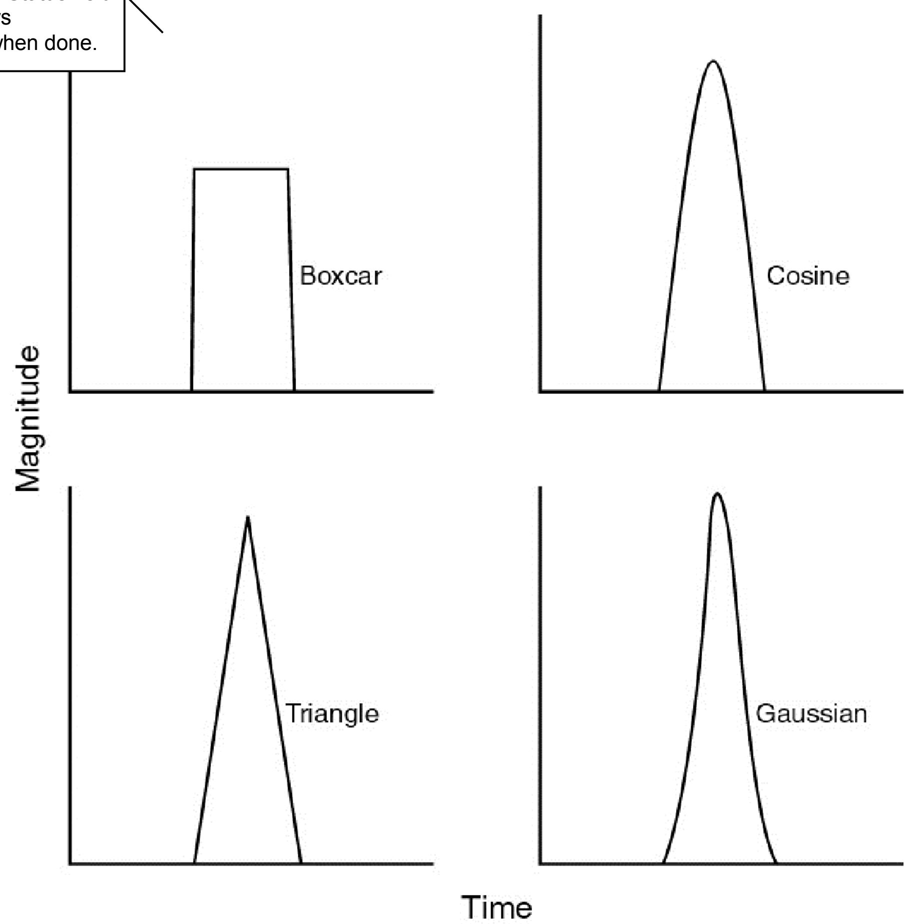
- Window Filter provides four window **shapes**: boxcar, cosine, triangle, and Gaussian.
- The minimum window **length** is 1 scan, and the maximum is 511 scans. Window length must be an odd number, so that the window has a center point. If a window length is specified as an even number, Window Filter automatically adds 1 to make the length odd.

The window filter calculates a weighted average of data values about a center point, using the following transfer function:

$$y(n) = \sum_{k=-L/2}^{L/2} w(k)\, x(n-k)$$

The figure below shows the impulse response of each of the four filter types for a filter of length 17 scans. The impulse response of a filter is obtained by filtering a data set that has zeros everywhere except one data value that is set to 1.

Begin processing data. Status field on File Setup tab shows *Processing complete* when done.



Note:
In the window filter equations:
- L = window length in scans, (always an odd number)
- n = window index, -L/2 to +L/2, with 0 the center point of the window
- w(n) = set of window weights

The window filtering process is similar for all filter types:
1. Filter weights are calculated (see the equations below).
2. Filter weights are normalized to sum to 1.
   - When a bad data point is encountered (scan marked with *badflag* if *exclude scans marked bad* was selected **or** data value marked with *badflag*), the weights are renormalized, excluding the filter element that would operate on the bad data point.

Boxcar Filter

$$w(n) = \frac{1}{L} \quad \textit{for } n = -\frac{L-1}{2} \,..\, \frac{L-1}{2}$$

Cosine Filter

$$w(n) = 1 \quad \textit{for } n = 0$$

$$w(n) = \cos \frac{n \times \pi}{L+1} \quad \textit{for } n = -\frac{L-1}{2} \,..\, {-1}, 1 \,..\, \frac{L-1}{2}$$

Triangle Filter

$$w(n) = 1 \quad \textit{for } n = 0$$

$$w(n) = \frac{|n|}{K} \quad \textit{for } n = -\frac{L-1}{2} \,..\, {-1}, 1 \,..\, \frac{L-1}{2}$$

$$\textit{where } K = \frac{L-1}{2} + 1$$

Gaussian Filter

$$\textit{phase} = \frac{\textit{offset}\ (\text{sec})}{\textit{sample interval}\ (\textit{sec})}$$

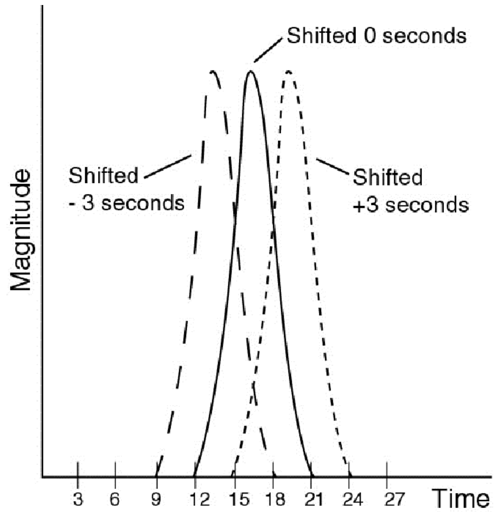
$$\textit{scale} = \log(2) \times \left(2 \times \frac{\textit{sample rate}}{\textit{half width}\ (\text{scans})}\right)^2$$

$$w(n) = e^{-\textit{phase} \times \textit{phase} \times \textit{scale}} \quad \textit{for } n = 0$$

$$w(n) = e^{-(n - \textit{phase})^2 \times \textit{scale}} \quad \textit{for } n = -\frac{L-1}{2} \,..\, {-1}, 1 \,..\, \frac{L-1}{2}$$

The Gaussian window has parameters of halfwidth (in scans) and offset (in time), in addition to window length (in scans). These extra parameters allow data to be filtered and shifted in time in one operation. Halfwidth determines the width of the Gaussian curve. A window length of 9 and halfwidth of 4 produces a set of filter weights that fills the window. A window length of 17 and halfwidth of 4 produces a set of filter weights that fills only half the window. If the filter weights do not fill the window, the offset parameter may be used to shift the weights within the window without clipping the edge of the Gaussian curve.

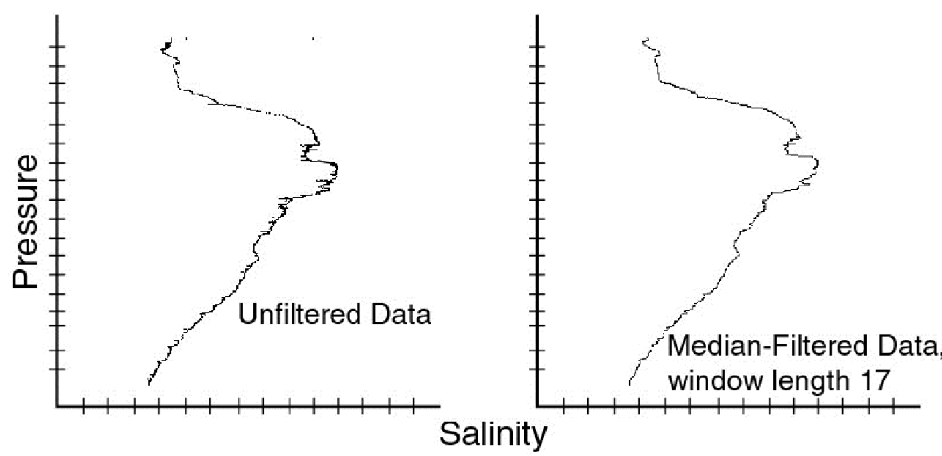
Example: Window length is 33 scans and halfwidth is 4 scans. Offset is -3 seconds in left curve, 0 in middle curve, and +3 seconds in right curve.



Note that the window length in the example is larger than the halfwidth. This allows the complete Gaussian curve to be expressed in the window when the offset parameter shifts the curve forward or backward in time. If the halfwidth was larger, the trailing edge of the -3 second offset curve would be truncated and the leading edge of the +3 second curve would be truncated. The offset parameter moves the Gaussian shape of the window weights forward or backward in time. Since the weighted average is calculated for a data value in the center of the window, this has the effect of shifting the data that the filter is operating on forward or backward in time relative to the other data in the file. This capability allows filtering and time shifting to be done in one step.

Median Filter: Description

The median filter is not a smoothing filter in the same sense as the window filters described above. Median filtering is most useful in spike removal. A median value is determined for a specified window, and the data value at the window's center point is replaced by the median value.



Window Filter has the following /x parameter when run from the Command Line Options dialog box, from the command line, or with batch file processing:

/x Parameter	Description
/xwfilter:diff	Output difference between original and filtered value instead of outputting filtered value.

See *Appendix I: Command Line Options, Command Line Operation, and Batch File Processing* for details on using parameters.

Window Filter adds the following to the data file header:

Label	Description
Wfilter_date	Date and time that module was run.
Wfilter_in	Input .cnv converted data file.
Wfilter_excl_ bad_scans	If yes, values in scans marked with *badflag* in Loop Edit will not be used.
Wfilter_action	Data channel identifier, filter type, filter parameters.