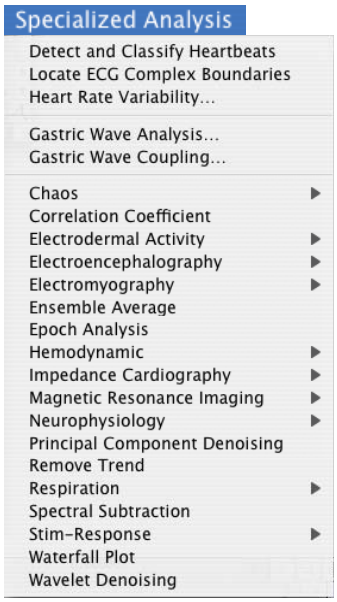
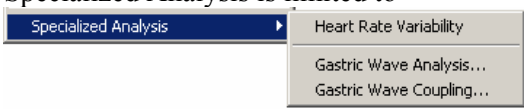
# Appendix G— Specialized Analysis in Acq*Knowledge* 3.9




The Specialized Analysis package includes comprehensive analysis tools to automate analysis to save hours (or days!) of processing time and standardize interpretation of results. Analyze data collected on MP Systems running on Windows/PC or Mac OS X.

*Windows* Specialized Analysis is limited to




For other Specialized Analysis functions offered on Mac, please read about File Portability (page 393).

*Mac OS X* Acq*Knowledge* 3.9 includes a courtesy copy of the Specialized Analysis Package under the Transform menu.

An analysis package is a bundle of transformations created to assist with analysis in a specific area of research.

A classifier is a special-purpose transformation that defines events at well-known points of interest on standard waveforms, such as the ECG wave boundary classifier and the QRS beat detector and arrhythmia detector.

The Specialized Analysis package includes the following Analysis Packages and Classifiers:

Detect and Classify Heartbeats
Locate ECG Complex Boundaries
Heart Rate Variability
Gastric Wave Analysis
Gastric Wave Coupling
Chaos Analysis
- Detrended Fluctuation Analysis
- Optimal Embedding Dimension
- Optimal Time Delay
- Plot Attractor

Correlation Coefficient
Electrodermal Activity
- Derive Phasic EDA from Tonic
- Event-related EDA Analysis
- Locate SCRs
- Preferences: Output Display Format; Phasic EDA Construction Method: Smoothing Baseline Removal or High Pass Filter

Electroencephalography
- Compute Approximate Entropy
- Delta Power Analysis
- Derive Alpha-RMS
- Derive EEG Frequency Bands
- EEG Frequency Analysis
- Remove EOG Artifacts
- Preferences: Output Display Format

Electromyography
- Derive Average Rectified EMG
- Derive Integrated EMG
- Derive Root Mean Square EMG
- EMG Frequency & Power Analysis
- Locate Muscle Activation
- Preferences: Output Display Format

Ensemble Average
Epoch Analysis
Hemodynamic Analysis
- Classifiers: ABP; LVP; MAP
- Arterial Blood Pressure
- ECG Interval Extraction

Hemodynamic Analysis *continued*
- Left Ventricular Blood Pressure
- Monophasic Action Potential
- Respiratory Sinus Arrhythmia
- Preferences: Output Display Format; LVDEP Location Method; dP/dt pk-pk %; MAP Plateau Location Method; dP/dt MAP pk-pk %

Impedance Cardiography Analysis
- Body Surface Area
- ICG Analysis
- PEP Pre-ejection Period
- dZ/dt Derive from Raw Z
- dZ/dt Classifier: B, C, X, Y, and O Points
- dZ/dt Remove Motion Artifacts
- Ideal Body Weight
- VEPT
- Preferences: Output Display Format; C-, B-, and X-Point Location; Stroke Volume Calculation Method; Body Measurement Units; Body Surface Area Method; Ideal Weight Estimation Method; dZ/dt Max Method

Magnetic Resonance Imaging
- Artifact Frequency Removal
- Artifact Projection Removal

Median Filter Artifact Removal
- Signal Blanking

Neurophysiology
- Amplitude Histograms
- Average Action Potentials
- Generate Spike Trains
- Find Overlapping Spike Episodes
- Set Episode Width & Offset
- Classify Spikes
- Dwell Time Histograms
- Locate Spike Episodes
- Preferences: Detect Spike; Default Episode Width; Default Episode Offset; Default # of Spike Classes

Principal Component Denoising
Remove Trend
Respiration
- Compliance & Resistance
- Penh Analysis
- Pulmonary Airflow

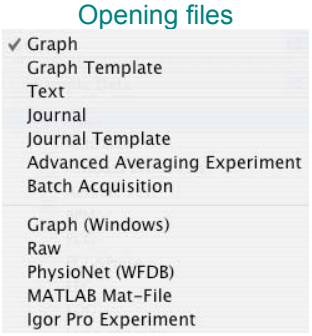
Spectral Subtraction
Stim-Response
- Digital Input to Stim Events
- Stim-Response Analysis

Waterfall Plot
Wavelet Denoising

Acq*Knowledge* File Portability Windows←→Mac OS 10.3 or higher

Use Specialized Analysis to analyze Acq*Knowledge* data files collected on MP Systems running on Windows/PC or Mac OS X. Using Acq*Knowledge* 3.9 for Specialized Analysis allows you to open/save the following file formats:

## Opening files for Specialized Analysis



Acq*Knowledge* 3.9

The default file formats (Graph and .ACQ) are referred to as "Acq*Knowledge*" files. The Acq*Knowledge* file format is the standard way of displaying waveforms in Acq*Knowledge*. These files are stored in a compact format that retains information about how the data was collected (i.e., for how long and at what rate) and takes relatively little time to read in (compared to text files, for instance). Acq*Knowledge* files are editable and can be modified and saved, or exported to other formats using the Save as command.

### File Compatibility

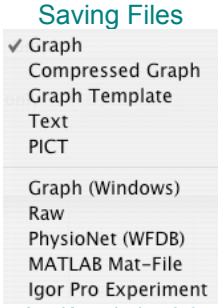
Windows Acq*Knowledge* 3.8 cannot open Mac Acq*Knowledge* 3.9 files.

Mac Acq*Knowledge* 3.9 can open Windows Acq*Knowledge* 3.8 or later files.

- Mac Acq*Knowledge* 3.9 can open and create PC-compatible Graph (*.acq) and Graph Template (*.gtl) files. Variable sampling rate information and hardware settings are retained, and Journals can be read from and written to PC files.

Files must end on a multiple of the lowest channel sampling rate to be fully PC compatible.

## Saving files after Specialized Analysis



Acq*Knowledge* 3.9

The default file format for the File>Save as command is to save files as an Acq*Knowledge* file. Selecting Graph (MPWS) or .ACQ (MPWSW) from the popup menu in the Save As dialog box will save a file as an Acq*Knowledge* file, which is designed to be as compact as possible. These files can only be opened by Acq*Knowledge*, but data can be exported to other formats once it has been read in.

The Options button generates a dialog box that allows you to save only a portion of your file. When the "Selected Section only" option is enabled, only the data that has been selected with the I-beam tool will be saved. This option saves the selected area to another file and does not affect the current file that you are working in.

### File Compatibility

Windows Acq*Knowledge* 3.9 files can be opened with Mac Acq*Knowledge* 3.9, but some advanced features may not transfer.

Mac Acq*Knowledge* 3.9 can save as "Graph (Windows)" files, but it saves in Windows Acq*Knowledge* 3.7.1 format. In this earlier format, all data is retained, but newer Acq*Knowledge* features (like dual stimulation, data views, embedded archives, etc.) are lost along with any settings specific to Mac Acq*Knowledge* (like events, adaptive scaling settings, etc.).

- Mac Acq*Knowledge* 3.9 can save PC-compatible Graph (*.acq) and Graph Template (*.gtl) files. Variable sampling rate information and hardware settings are retained, and Journals can be read from and written to PC files. Choose the format "Graph (Windows)" to create PC-compatible files.

The Mac version does not save PC GLP files or compressed PC files.

Files must end on a multiple of the lowest channel sampling rate to be fully PC compatible.

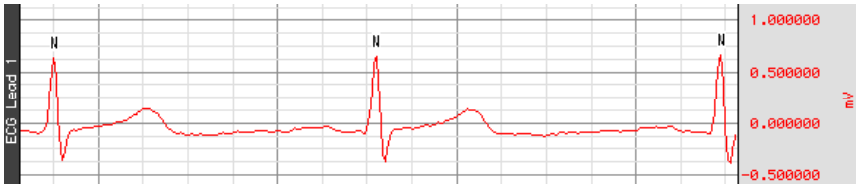
Excel Spreadsheet Export Acq*Knowledge* 3.9.1 Mac only—The Specialized Analysis tools have been updated to automatically export their results to an Excel spreadsheet if desired. The spreadsheet contents mirror the tabular Journal text output. All the spreadsheets are saved as temporary files, so they need to be re-saved in order to be saved permanently.

- Also available for File > Save As, File > Save Journal Text As, and Find All Cycles journal.

*Note* Specialized Analysis scripts are complex and undo may not function for all steps.

Some of the specialized algorithms are very complex and processor intensive, so they may take a long (even *very* long) time to return a result.

## Detect and Classify Heartbeats

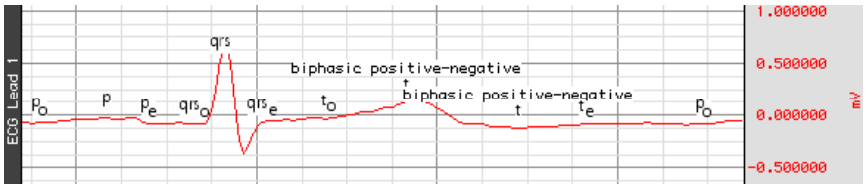



This robust QRS detector is tuned for human ECG Lead II signals. It attempts to locate QRS complexes and places an event near the center of each QRS complex to identify the type of heartbeat event:

- Normal The beat was recognizable as a valid heartbeat falling in a human heartbeat rate.
- PVC The beat was shorter than the beats around it and may be a pre-ventricular contraction. These events can be found in the "Hemodynamic > Beats" submenu of the event type listing.
- Unknown The beat wasn't recognizable as a valid heartbeat. This may occur on the first beat prior to the QRS detector locking onto the signal. It may also occur if tracking is lost due to changes in signal quality.

The Cycle/Peak detector may be used with these events to perform further cardiac analysis.

## Locate ECG Complex Boundaries




This classifier performs ECG waveform boundary detection for human ECG Lead II signals. It will attempt to locate the boundaries of the QRS, T, and P wave and will define events for each individual complex. It will attempt to insert the following events; all of these complex boundaries can be found in the "Hemodynamic > ECG Complexes" submenu of the Event Type listing.

| Wave | Type | Event Placement & Description |
|---|---|---|
| QRS | Onset | Before the beginning of the Q wave |
| | Peak | At the top of the R wave |
| | End | After the end of the S wave |
| T-wave | Onset | At the onset of T |
| | Peak | At the peak of the T wave |
| | | *Note:* This may not be a positive peak if the T-wave is inverted. If the T-wave seems to be bi-phasic, two T-wave events will be inserted and the event description will indicate that the T-wave is bi-phasic. |
| | End | At the end of T |
| P-wave | Onset | At the onset of P |
| | Peak | At the top of the P wave |
| | | *Note*: This may not be the absolute maximum, but rather the likely center of P. |
| | End | At the end of P |

The Cycle/Peak detector may be used with these events to perform further cardiac analysis.

## Heart Rate Variability

Recording good data is essential for performing HRV analysis. The protocol for data acquisition, filtering, artifact detection and correction in Application Note 233 results in great improvements in HRV analysis.

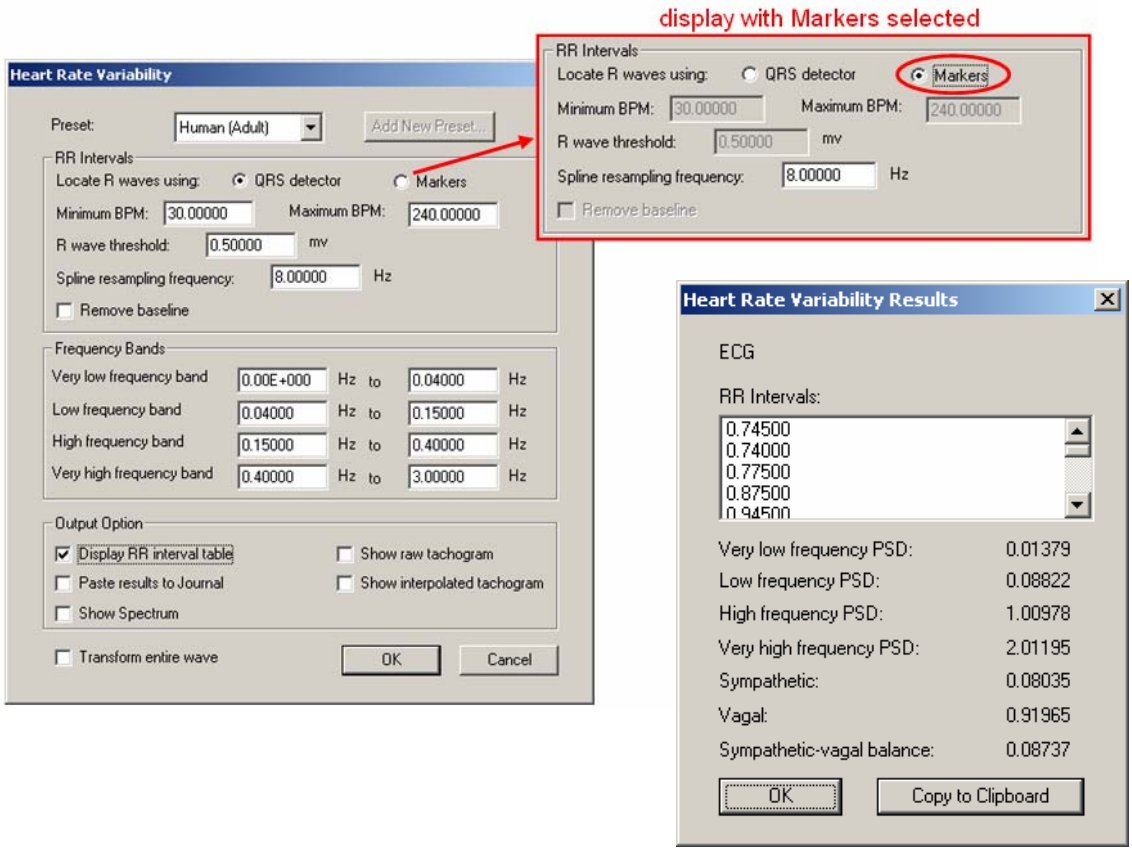
> "Results reveal that even a single heart period artifact, occurring within a 2-min recording epoch, can lead to errors of estimate heart period variability that are considerably larger than typical effect sizes in psychophysiological studies." —Berntson & Stowell, 199

- See **Application Note 233 Heart Rate Variability—Preparing Data for Analysis Using Acq*Knowledge* (online at www.biopac.com)**

The note explains how to optimize ECG R-R interval data for Heart Rate Variability studies by using a template matching approach. It also explains how to identify erroneous R-R interval values caused by signal artifact and shows methods for correcting the errors by using the tools in the Acq*Knowledge* software. The note explains how to:

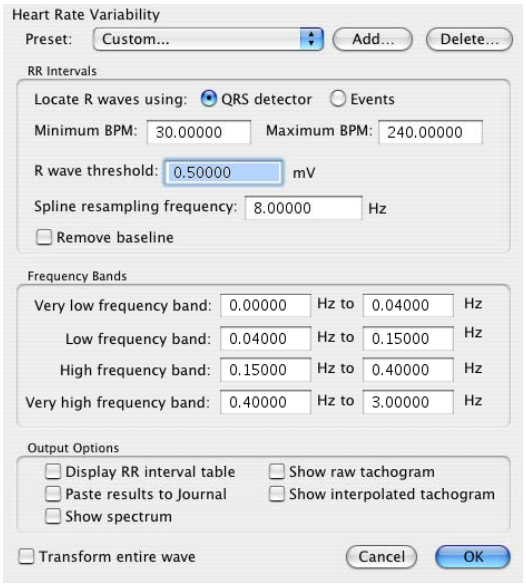
A. Record good ECG data
B. Prepare data for the tachogram
   1. Filter the ECG data
   2. Transform the data using Template Correlation function
C. Create a tachogram
D. Identify problems with the tachogram data
E. Correct tachogram data

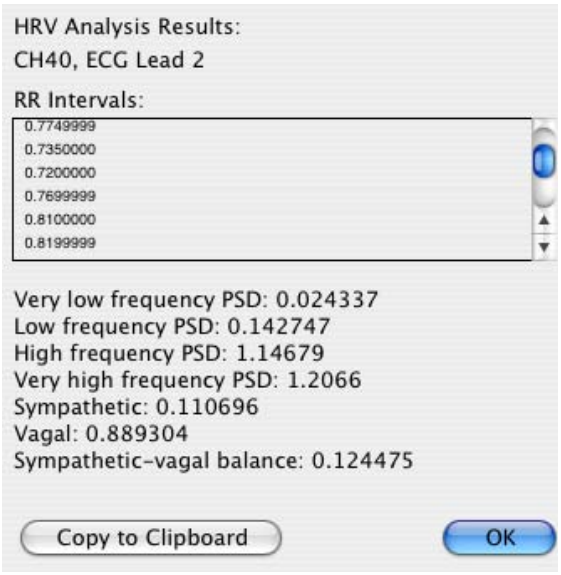
### Windows




Windows Results

## Mac

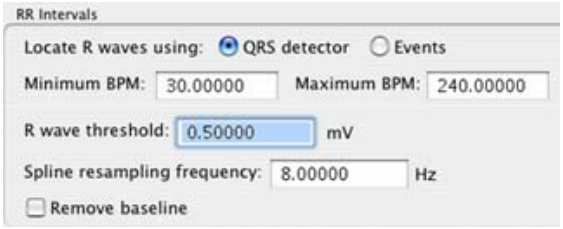


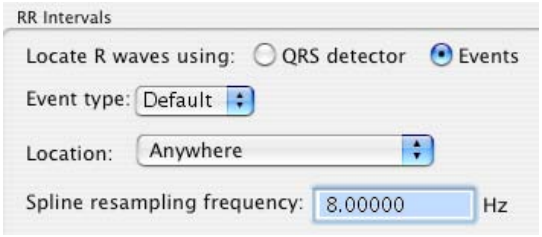




Heart rate variability is the examination of physiological rythms that exist in the beat-to-beat interval of a cardiac signal. Heart rate variability assists in performing frequency domain analysis of human ECG Lead II data to extract standard HRV measures. The Acq*Knowledge* 3.9 algorithm conforms to the frequency domain algorithm guidelines as published by the European Heart Journal. HRV processing in Acq*Knowledge* consists of three stages:

1. The RR intervals are extracted for the ECG signal.
   - A modified Pan-Tompkins QRS detector is used.
2. The RR intervals are re-sampled to a continuous sampling rate in order to extract frequency information.
   - Cubic-spline interpolation is used to generate this continuous time-domain representation of the RR intervals.
3. The frequency information is extracted from the RR intervals and analyzed to produce standard ratios. Power sums are reported in units of $sec^2$.
   - A Welch periodogram is used to generate the Power Spectral Density (equivalent to Transform > Power Spectral Density).

RR intervals—Select a method to locate R waves




### QRS detector

The HRV analysis QRS detector uses a modified Pan-Tompkins algorithm. The algorithm normalizes the ECG data to 1 whereby the peak amplitude of the highest R-wave represents 1. The default threshold level of .5 should place the threshold in the middle of the R-wave. If the R-wave amplitude varies a lot, it might be necessary to adjust the threshold level.

Pan J and Tompkins WJ. A Real-Time QRS Detection Algorithm. *IEEE Transactions on Biomedical Engineering* ***32***(3):230-236, 1985.

The standard settings should function on a wide range of data sets, but you can adjust the thresholds for your particular data set. Use the tachogram output to examine the output of the QRS detector.

- *Spline resampling frequency*—For highest accuracy, set to no less than twice the topmost frequency of the very high frequency band.

Frequency Bands—Adjust the boundaries of the frequency analysis. They are preset to the frequency ranges recommended by the *European Heart Journal*. Output of derived parameters is presented in a Sheet and may also be pasted as text to the Journal.

Output Options— Create standard result presentation graphs or assess performance of the HRV algorithm. These Options allow access to intermediate computation data for algorithm validation and/or measurements.

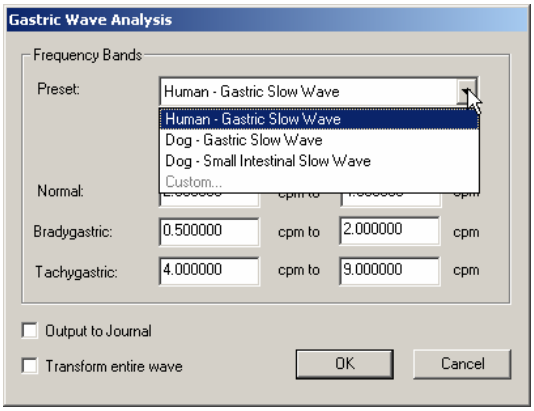
- Spectrum—Displays the power spectrum density (PSD) estimation from which the PSD summations and sympathetic/vagal ratios are computed.
- Raw tachogram—Plots the raw R-R intervals found by the QRS detector. Perform statistical HRV measures on the R-R intervals without exporting the textual R-R table to excel.
- Interpolated tachogram—Plots the resampled R-R intervals after cubic spline interpolation is applied and extracts the PSD from this data.
- Presets—The preset menu can be used to save a variety of HRV settings, including: beat detection parameters, spline resampling frequency, and frequency band ranges. Choose a preset from the popup menu to apply its settings. To construct a new preset with the currently displayed settings, choose Add New Preset. A default preset for adult human subjects is supplied.

### Events

Located using events in the channel of data to be analyzed. This assumes a single event is placed at each R-wave peak and that all of the R-peak events are of the same event type. When using events, the buit-in QRS detector is not used; the exact positioning between the events on the channel is used to extract the RR intervals.

By using events, it is possible to use other QRS detectors within Acq*Knowledge* for performing HRV analysis. It is also possible to apply spectral HRV-style analysis to data in other domains as long as intervals can be reduced to events.

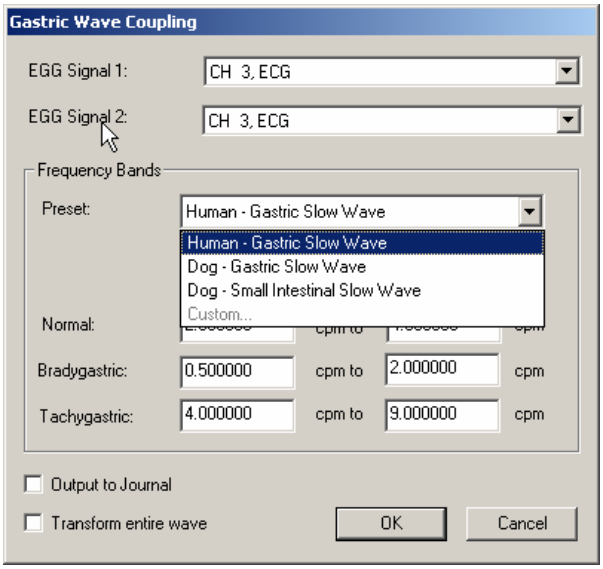
## Gastric Wave Analysis

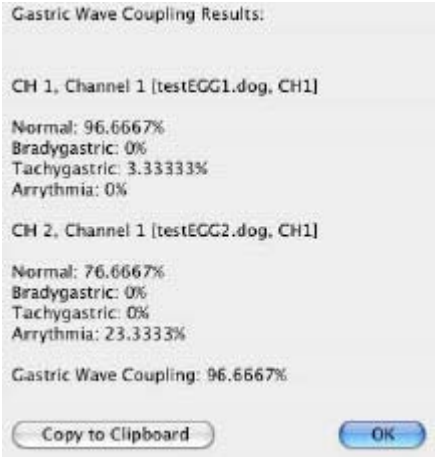




Gastric Wave Analysis uses autoregressive time-frequency analysis to determine the classifications of gastric waves present in an EGG signal. The single wave analysis determines the percentage of gastric waves that fall within the frequency bands corresponding to normal, bradygastric, and tachygastric waves. The analysis also indicates the percentage of waves that fall outside of these boundaries and are arrhythmias. The frequency bands are expressed in units of "contractions per minute" and may be adjusted by the user. Presets for commonly used subject and wave types are predefined; you may extend these presets with your own.

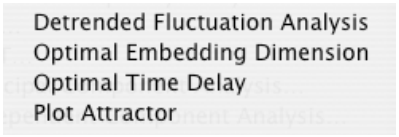
## Gastric Wave Coupling



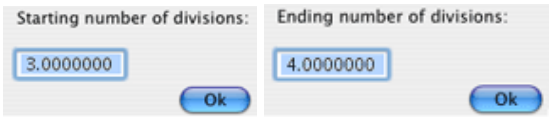

Gastric Wave Coupling takes two EGG signals and uses autoregressive techniques to classify the contractions in those signals according to user-configurable frequency bands (similar to single channel Gastric Wave Analysis). In addition to providing classification information for the two signals, Gastric Wave Coupling provides a measure of the percentage of coupling between the two signals—this measure that can be used to determine the amount of slow-wave propagation across the stomach.

## Chaos Analysis




The "Chaos" analysis package assists the user in exploring the chaotic nature of data, including measurement selection and visualization of time domain attractors in the data.

### *Detrended Fluctuation Analysis*



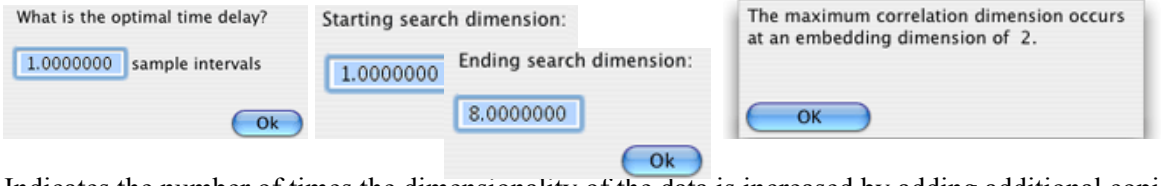

Modified root mean square analysis, useful for evaluating self-similarity in a long-term, non-stationary data series. Source data is mean-adjusted and then integrated; it is then split up into *n* segments of equal length, and in each segment, via linear regression, the best fit least squares line is computed. For a particular value of *n* and a number of samples *N*, the characteristic fluctuation of the piecewise linear fit $y_n$. is defined as:

$$F(n)=\sqrt{\frac{1}{N}\sum_{k=1}^{N}[y(k)-y_n(k)]^2}$$

*F(n)* is evaluated over a user-specified range for the number of divisions. *n* will equal the total length divided by the number of divisions. A log-log plot of the interval width *n* in samples versus the corresponding value of *F(n)* will be created. If a linear relationship appears to exist in this graph, then the source signal displays some form of self-similarity. The slope of the line in this graph is related to the scaling exponent.

➲ For more information on Detrended Fluctuation Analysis, see http://www.physionet.org/physiotools/dfa/

### *Optimal Embedding Dimension*

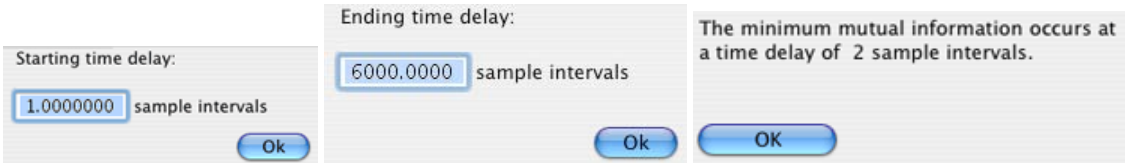



Indicates the number of times the dimensionality of the data is increased by adding additional copies of the data. Many of the fractal measurements take an embedding dimension parameter. Increasing the dimensionality of the data may improve the quality of the results. In general, embedding dimensions should always be less than 8.

After the most relevant time delay for the data has been selected, Optimal Embedding Dimension assists in choosing the embedding dimension that appears to give the most accurate results. The embedding dimension is chosen to be the earliest dimension in the search range where the fractal correlation dimension measure reaches a local maximum. This indicates the lowest dimension where the data has the potential to exhibit the most self-similarity.

- Since real data may not be fractal in nature, there may be no local maximum for the embedding dimension. In this case, it is not possible to determine the optimal dimension.

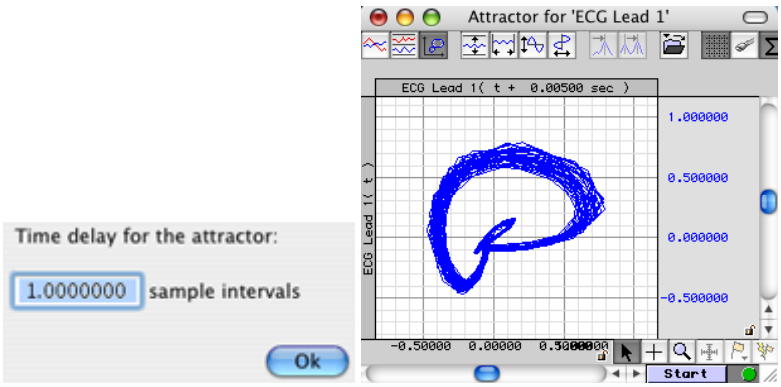
### *Optimal Time Delay*




Assists in picking a time delay that is most relevant for the data. It runs through and locates the earliest time delay in the specified interval range where the mutual information measurement reaches a local minimum. Optimizing the time delay in this fashion picks the shortest delay where the signal exhibits the most independence with respect to its time-delayed version.

The new fractal dimension and other chaos-related measurements operate on a single channel of data. During the process of extracting these measures, a signal is compared with a time-delayed version of itself to examine the patterns in dynamics of the data. These measures take a fixed time delay setting to perform this analysis. The Optimal Time Delay transformation can be used to choose the best value for the parameter.

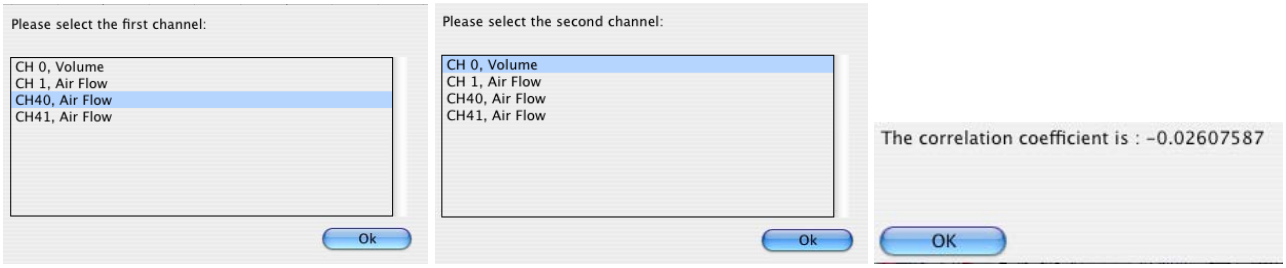
### *Plot Attractor*




Assists in constructing X/Y plots for the attractors of time delayed data. By visually examining the shape of the attractor at a given time delay, To develop an intuitive sense for the underlying nature of the data and the dynamics of the system.

Plot Attractor functions on the active channel of the graph. It prompts the user for a time delay and then constructs a new graph window with an X/Y plot of the attractor of the original signal against the time delayed version of the signal. It does not perform any additional computation aside from assisting in the setup and configuration of the attractor plot.

## Correlation Coefficient




The *correlation coefficient* is a statistical measure related to the degree of variance or covariance between two data series. Given two data series *x* and *y* of length *n*, the correlation coefficient *r* is given by the formula:

$$r = \frac{n\sum xy - \sum x \sum y}{\sqrt{\left[n\sum x^2 - (\sum x)^2\right]\left[n\sum y^2 - (\sum y)^2\right]}}$$

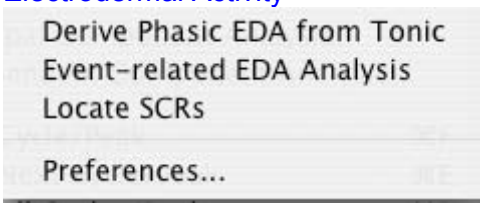
(see http://mathworld.wolfram.com/CorrelationCoefficient.html)

The square of the correlation coefficient can be used to determine the proportion of variance in common between the two signals. As the square gets closer to 1, the signals are a better statistical match for each other.

To derive the correlation coefficient, two channels of data are compared against each other.

- the channels must have the same length
- the channels must have the same waveform sampling rate
- all of the data of the entire graph for the two channels will be used to compute the correlation coefficient.

## Electrodermal Activity




### *Definitions*

The prompts and analysis results of the EDA analysis package use the following terminology and units:

µmho—the unit abbreviation for microSiemens, used in channel labels and analysis results.

EDA (Electrodermal Activity)— the general area of skin conductance signals. Sometimes referred to by the older term "galvanic skin response."

Tonic EDA—continuous data acquired from an EDA electrode that includes all baseline offset. Sometimes referred to as "skin conductance level." Averaging the tonic EDA over a specific period of time results in the average skin conductance over an interval. Tonic EDA is recorded using BIOPAC equipment with the highpass filtering set to off (DC mode).

Phasic EDA—a continuous signal indicative of localized changes in the tonic EDA signal. Sometimes referred to as "continuous skin conductance response." Phasic EDA can be thought of as AC coupled tonic EDA. The EDA analysis package offers multiple ways of constructing phasic EDA including smoothing and highpass filtering. The EDA analysis package performs the majority of its analysis on tonic EDA signals, so if phasic EDA is being recorded directly it is recommended that a second channel be used to record tonic EDA.

Skin Conductance Response (SCR)—an individual localized change in the tonic EDA signal. An SCR may occur in response to a stimulus or may occur spontaneously. In general, there are multiple SCRs present in a tonic EDA signal and they can be detected as deflections from the localized baseline.

### *Derive Phasic EDA from Tonic*

Given a tonic EDA signal, this transformation uses baseline smoothing or highpass filtering (the method currently set in Preferences) to construct a new Phasic EDA channel in the graph containing the estimate of the phasic EDA.

### *Event-related EDA Analysis*

Many studies examine how EDA changes in response to certain stimuli. By examining changes in EDA, it is possible to gauge a subject's psychological response to a certain event. The Event-related EDA Analysis transformation script assists in the extraction of EDA measures that are linked to specific stimuli.

In order to use the Event-related EDA Analysis transformation script, an event must be defined in the graph at the location of the delivery of each stimulus. This event may be defined using the Event Tool, hotkey insertion during acquisition, or any other method of defining events. Stimulus delivery events are located by event type and (optionally) by specific channel of the graph. All of the stimulus delivery locations to be extracted must have the same event type (e.g. "Flag"). To analyze multiple different event types, the transformation script must be executed multiple times.

In addition to the stimulus delivery events, a tonic EDA signal must be present in the graph. If the tonic EDA signal does not already have SCR events defined on it, SCR events will be automatically constructed on the channel in the same fashion as the Locate SCRs transformation script.

Each stimulus delivery event is paired with the closest SCR event. SCRs that correspond to a stimulus delivery are known as "specific SCRs." SCRs generally occur within a certain timeframe after stimuli. The transformation takes a maximum allowable separation interval between the stimulus and SCR response. If the closest SCR to a stimulus is farther away than this time interval, it is not assumed to be a response to the stimulus. It may be a response to a later stimulus or it may be a non-specific SCR that occurred spontaneously.

For each specific SCR that is paired with a stimulus delivery event, the following measures are extracted:

| *Name* | *Abbrev.* | *Description* | *Units* |
|---|---|---|---|

| Name | Abbrev. | Description | Units |
|---|---|---|---|
| Stimulus Delivery Time | Stim Time | The time within the recording where the stimulus delivery event was located. | seconds |
| Skin Conductance Level | SCL | Amplitude of the tonic EDA signal at the time when the stimulus was delivered. | umho |
| Response Latency | Latency | Time separating the stimulus delivery from the onset time of the corresponding SCR.<br>This latency will always be less than the maximum allowable latency specified as a parameter for the analysis. | seconds |
| SCR Amplitude | SCR Amplitude | Height of the corresponding SCR as determined by the change in the tonic EDA from the time of SCR onset to the maximum tonic EDA amplitude achieved during the SCA: $\left\|EDA(t_{max}) - EDA(t_{onset})\right\|$ | umho |
| SCR Rise Time | SCR Rise Time | Time taken for the tonic EDA to reach its maximum value within the SCR: $t_{max} - t_{onset}$ | seconds |

If text output is enabled, the average value of SCL, Latency, SCR Amplitude, and SCR Rise Time will be included as the final row of the table.

In addition to the above measures extracted for each specific SCR, the analysis performs rate extractions for specific and non-specific SCRs. By examining how the rate of SCR occurrences changes, long-term experimental trends can be investigated. This analysis is placed into a second set of waveforms (or a second table for text and Excel output).

A fixed width window is specified as the "SCR count interval width" when performing the analysis. The entire recording is split up into fixed-width epochs of this granularity with the first epoch aligned at the start of the recording. For each fixed-width epoch, the following are extracted:

| Name | Abbrev. | Description | Units |
|---|---|---|---|
| Epoch Start Time | Start Time | Time location in the recording of the start of the epoch being examined. | seconds |
| Specific SCR Rate | SRR | Frequency of the occurrences of specific SCRs within the epoch. Specific SCRs are those SCRs that were successfully matched to a corresponding stimulus delivery event. | Hz |
| Non-specific SCR Rate | NS.SRR | Frequency of the occurrences of non-specific SCRs within the epoch. These are SCRs that occur spontaneously and are not paired with any known stimulus. | Hz |

### *Locate SCRs*

Given a tonic EDA, this transformation defines event triplets for each skin conductive response in the tonic EDA. SCR location is a two stage process.

1. First, all potential SCR occurrances are located on the signal.
2. Second, all potential SCR occurrances that are not large enough are rejected.

Potential SCR occurrences are detected by performing thresholding positive peak detection on the phasic EDA signal (using H and P as set via Preferences):

1. Given a detection threshold H (expressed in umho), search for a positive threshold crossing in the phasic EDA signal. This position, $t_{onset}$ is recorded as the start of the potential SCR.

2. Starting at $t_{onset}$, continue examining the phasic EDA until the first negative threshold crossing of 0 umho occurs. This position, $t_{end}$ is recorded as the end of the potential SCR.
3. Return to step 1 to continue searching for more potential SCRs occurring after $t_{end}$.

After all of the potential SCRs have been located, the set of valid SCRs is constructed as follows:

1. Determine the overall maximum amplitude of the phasic EDA signal within all potential SCRs.
2. Given a percentage P, construct a threshold level T of P percent of the overall maximum phasic EDA signal value located in step 1.
3. Examine each potential SCR. Find the maximum phasic EDA $m$ within $[t_{onset}, t_{end}]$. If $m < t$, discard the potential SCR. If $m \geq t$, mark the potential SCR as a valid SCR.

This transformation requires a tonic EDA signal. If a phasic EDA has already been constructed for this tonic EDA, it may be used; otherwise, the transformation will create a phasic EDA automatically according to the settings in the Preferences.

If the tonic EDA channel chosen for analysis already has SCR events defined on it, the SCR events will be replaced with the newly detected SCR events. No existing SCR events will be erased without a confirmation.

Once SCR events have been defined, they can be used in conjunction with the Cycle Detector for performing further data reduction. The "event count" measurement can be used to estimate SCR frequency during individual time ranges of the experiment.

### *Events on Tonic EDA*

After valid SCRs are located using the algorithm above, events are inserted into the graph that can allow for further data analysis around the SCR positions. Three events are defined on the tonic EDA waveform for each individual valid SCR:

1. "General > Waveform onset" event at the SCR onset time $t_{onset}$. This is the point where the threshold H was crossed in the phasic EDA.
2. "EDA > Skin conductance response" event at the time $t_{max}$ where the tonic EDA reaches its maximum value within the SCR (max in time range $[t_{onset}, t_{end}]$).
3. "General > Waveform end" event at the ending SCR time $t_{end}$. This is the point where the zero threshold was crossed in the phasic EDA.

Events for SCRs will always occur in the above triplets, in the order shown.

### *Preferences…*

The following preferences can be configured and will be applied to all options in the analysis package:

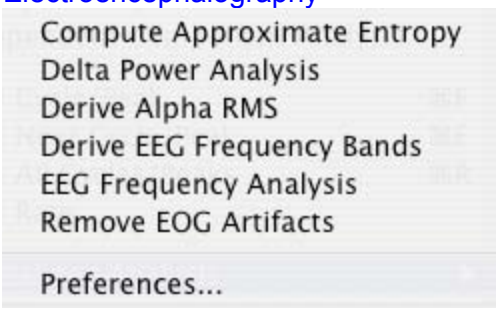
- Output: text, graph channels, or Excel
- Phasic EDA options: Highpass Filtering or Smoothing Baseline Removal
  - Highpass Filtering—Highpass filtering constructs phasic EDA by applying a digital IIR highpass filter (f = 0.05 Hz, Q = 0.707) to the tonic EDA signal. This highpass filter essentially AC couples the tonic EDA signal similar to using the highpass hardware filter available on the GSR100C module. When using highpass filtering, the first few seconds of the phasic EDA may not be centered around zero and will appear to contain invalid data.; this artifact is related to the long time constant of this filter and is expected.
  - Smoothing Baseline Removal—Smoothing baseline removal constructs phasic EDA by subtracting an estimate of the baseline conductance from the tonic EDA. The estimate of the baseline is generated using median value smoothing. This is more computationally intensive than highpass filtering but does not illustrate artifact at the start of the signal.

    Additionally, its sensitivity can be adjusted by varying the window size. The width of the window is customizable in the preferences.

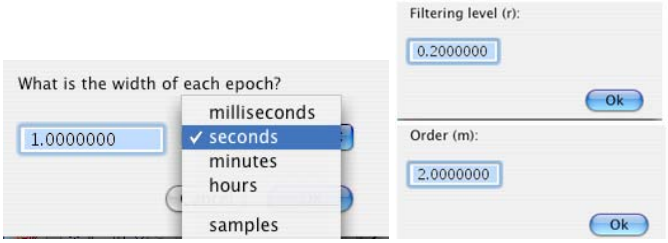
- Larger windows will decrease the sensitivity and show only large changes.
- Smaller windows will increase sensitivity and show more localized responses.

- SCR detection parameters: threshold detection level H and percentage P
  - The default values are H = 0.02 umho, P = 10.
  - Setting H to 0 and P to 10% will approximate the referenced SCR detection algorithm.
  - Setting P to 0% will retain all potential SCRs (none will be rejected in the second phase).

## Electroencephalography



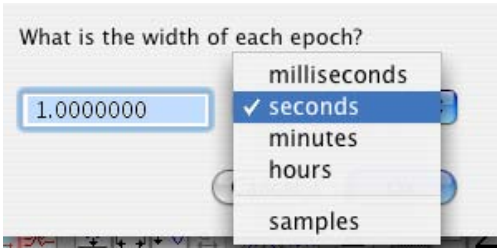

### *Compute Approximate Entropy*




*Approximate entropy* is a statistical measure that attempts to quantify the predictability of a data sequence. A perfectly predictable data series (such as a pure sine wave) has an approximate entropy of zero. Several studies are beginning to examine approximate entropy of EEG data and its relationship to external factors such as drugs and sleep states.

The Compute Approximate Entropy script divides an EEG signal into fixed-width epochs and computes the approximate entropy for each epoch. Derivation of the approximate entropy is a computationally intensive process and may take several minutes or hours to complete. To obtain only the sub-ranges of the EEG data, copy and paste the ranges into new graph windows to restrict the approximate entropy computations to that data range; the analysis is performed for all of the data in the graph window regardless of the selected area.

### *Delta Power Analysis*




*Delta power* is the total power of the EEG signal that occurs within the delta frequency band as configured in the Preferences. Delta power has been examined in a number of various EEG studies as an indicator of sleep/wakefulness and other conditions. By examining changes in the delta power, it may be possible to correlate delta power with effects of external factors.

The Delta Power Analysis script divides an EEG channel into fixed-width epochs. For each epoch, the power spectral density is computed and the total power within the delta frequency band is derived from the PSD. This delta power value is then placed into the graph or into the journal as specified by the output preferences.

Delta power can be measured from either a filtered or unfiltered EEG channel. To compute delta power for individual frequency bands, they must be derived prior to running the Delta Power Analysis script.

### *Derive Alpha RMS*

The Derive Alpha RMS script constructs a standard alpha RMS waveform from an alpha EEG signal. Alpha RMS is the windowed root mean square value of the signal using a window width of 0.25 seconds.

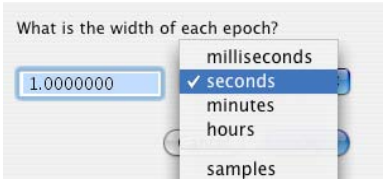
### *Derive EEG Frequency Bands*

The Derive EEG Frequency Bands script applies filtering to an unfiltered EEG lead signal to generate the following five standard EEG bands:

- Alpha
- Beta
- Theta
- Delta
- Gamma

The frequencies used for each band are taken from the analysis package preferences. Filtering is performed using IIR lowpass+highpass combinaton filters.

### *EEG Frequency Analysis*

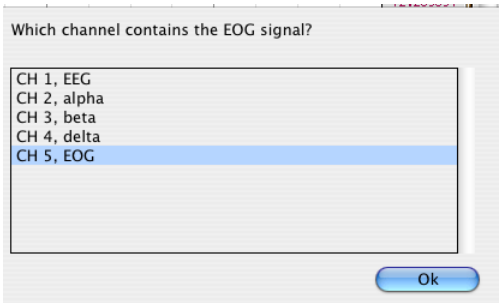


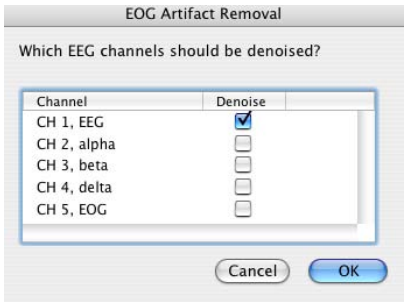

EEG may be characterized in terms of frequency and the power within specific frequency bands. The EEG Frequency Analysis script performs various feature extractions from EEG signals using FFT and other techniques to examine the power within the EEG signals. This analysis may be performed for multiple EEG leads simultaneously, allowing for either analysis of multiple leads or analysis of multiple EEG alpha, beta, theta, or delta bands from a single raw lead.

The EEG Frequency Analysis script divides the EEG signals into fixed-width time epochs. For each individual time epoch, Acq*Knowledge*'s Power Spectral Density function is used to estimate the power spectrum of that epoch using a Welch periodogram estimation method. From this PSD the following measures are extracted for each epoch:

| *Name* | *Abbrev.* | *Description* | *Units* |
|---|---|---|---|
| Mean Power | MeanP | The average power of the power spectrum within the epoch.<br>(Units Note: V will be replaced with the voltage units in which the EEG was recorded) | $\frac{V^2}{Hz}$ |
| Median Frequency | MedianF | Frequency at which 50% of the total power within the epoch is reached. | Hz |
| Mean Frequency | MeanF | Frequency at which the average power within the epoch is reached. | Hz |
| Spectral Edge | Spectral Edge | Frequency below which a user-specified percentage of the total power within the epoch is reached. This percentage can be set using "Preferences" and defaults to 90%. | Hz |
| Peak Frequency | PeakF | Frequency at which the maximum power occurs during the epoch. | Hz |

### *Remove EOG Artifacts*




Some EEG recordings involve subjects performing various visual tasks such as reading or watching video. Under these conditions, EEG may be susceptible to interference from the much stronger EOG signal arising from eye motion, particularly if EEG is recorded from near the front of the skull. Remove EOG Artifacts helps remove EOG interference from the EEG signals, recovering the EEG data for use in further analysis.

EOG removal is performed using a blind signal separation technique known as Independent Component Analysis. ICA is used to split up statistically independent signals that have been mixed together during recording. Since EOG is independent of EEG, ICA can be used to remove it.

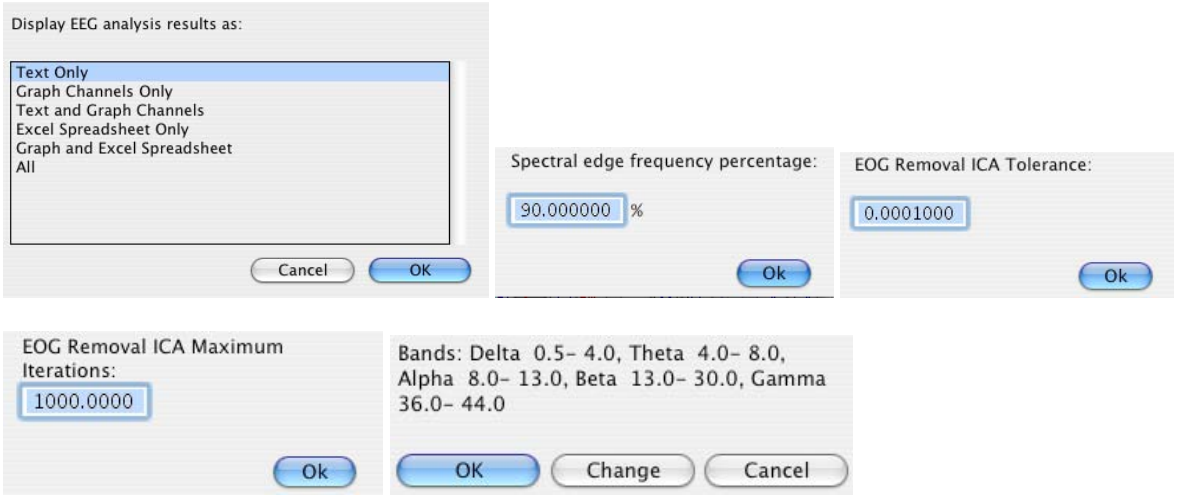
In order to use Remove EOG Artifacts, a distinct EOG signal must be acquired in addition to the EEG signals. The EOG signal is required to identify the components correlated to eye motion.

EOG artifact removal functions better when it is performed on multiple EEG leads simultaneously. Better results may be obtained by including EEG leads that do not exhibit EOG interference since the increased

number of leads allows for more fine-grained signal separation. Good results can be seen with as few as two EEG leads and one EOG lead. While this technique can be performed with a single EEG lead, the results will not be as dramatic.

*Note* ICA is a non-deterministic technique, so it may not be possible to automatically separate the signals for every EOG/EEG data set. For ICA to be successful, it may be necessary to fine-tune the parameters of the ICA search procedure to match the data, use a different electrode configuration, or use fewer or more leads.

*Preferences…*




Adjust the EOG ICA Tolerance level and the EOG ICA maximum number of iterations by accessing Transform > Specialized Analysis > Electroencephalography > Preferences. EOG ICA Tolerance is used as the termination condition of ICA signal separation. The EOG ICA maximum number of iterations is another termination condition of ICA signal separation and represents the maximum point at which the search is aborted. For more information on these settings, see the documentation for the Independent Component Analysis transformation.

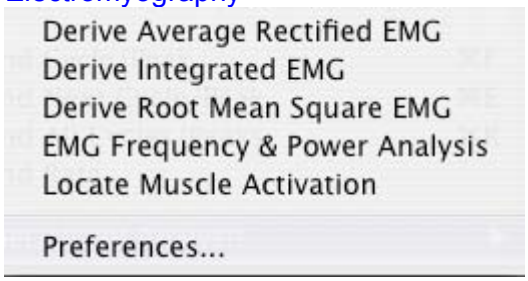
Because ICA is a stastical technique, any filtered data produced with Remove EOG Artifacts should be carefully verified against other information to ensure that the approximations produced via ICA represent information that is truly correlated to the expected ECG.

The spectral edge percentage indicates the cutoff percentage of the total power at which spectral edges will be placed. The default value is 90%.

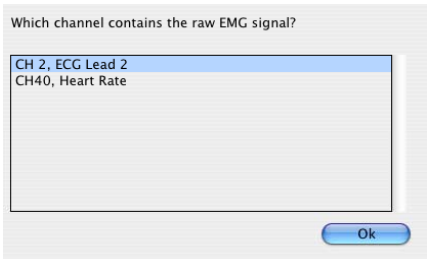
The frequency bands of alpha, beta, delta, and theta may be modified to match different analysis protocols. The default frequency ranges are:

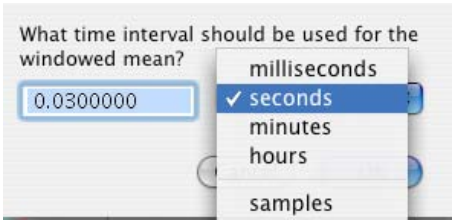
- Alpha—8 Hz-13 Hz
- Beta—13 Hz-30 Hz
- Delta—0.5 Hz-4 Hz
- Theta—4 Hz-8 Hz
- Gamma—36 Hz-44 Hz

## Electromyography




### *Derive Average Rectified EMG*



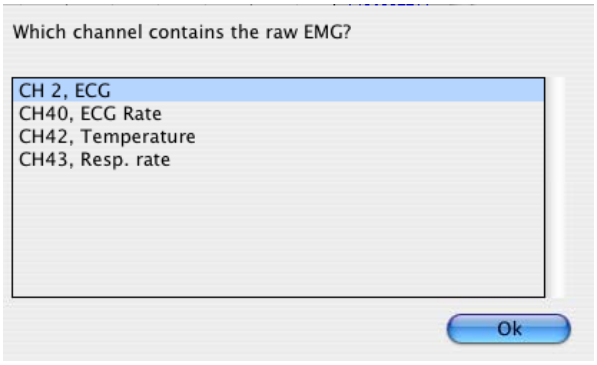



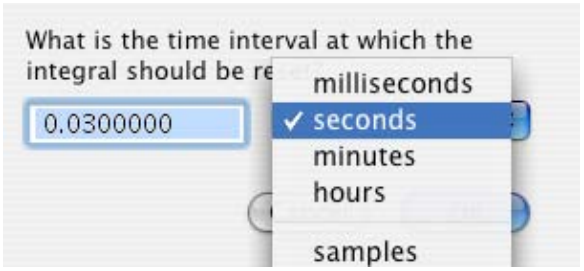

*Average rectified value* (ARV) is defined as a time windowed mean of the absolute value of the signal. ARV is one of the various processing methods used to construct derived signals from raw EMG data that can be useful for further analysis.

To perform ARV, a time window must be specified for the sliding mean. The default time window setting is 30 milliseconds, but this value can be adjusted depending on the desired amount of smoothing effects. It is advisable to closely examine results for time windows larger than 30 milliseconds as it is possible for delay to be introduced into the result.

The ARV is computed using the Integrate transformation with a Rectified Average over Samples configuration.

### *Derive Integrated EMG*

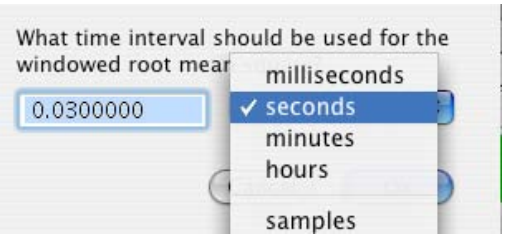



*Integrated EMG* (iEMG) is defined as the area under the curve of the rectified EMG signal, that is, the mathematical integral of the absolute value of the raw EMG signal. When the absolute value of the signal is taken, noise will make the mathematical integral have a constant increase. Integrated EMG splits up the signal into fixed-width timeslices and resets the integral at the start of each timeslice. To derive iEMG, the width of this timeslice must be specified. Similar to ARV, timeslices longer than 30 milliseconds may introduce delay into the result.

The integrated rectified EMG signal will appear like a “sawtooth” style wave. In addition to the true iEMG, this script will output a second waveform whose value is the maximum value of the iEMG signal in each timeslice. This Maximum iEMG is easier to interpret visually and approximates the envelope of the iEMG signal.

### *Derive Root Mean Square EMG*




*Root Mean Square EMG* (RMS EMG) is defined as the time windowed RMS value of the raw EMG. RMS is one of a number of methods used to produce waveforms that are more easily analyzable than the noisy raw EMG.

To construct the windowed RMS signal, a time window must be specified for the sliding mean. The default time window setting is 30 milliseconds, but this value can be adjusted depending on the desired amount of smoothing effects in the RMS EMG. It is advisable to closely examine results for time windows larger than 30 milliseconds as it is possible for delay to be introduced into the result.

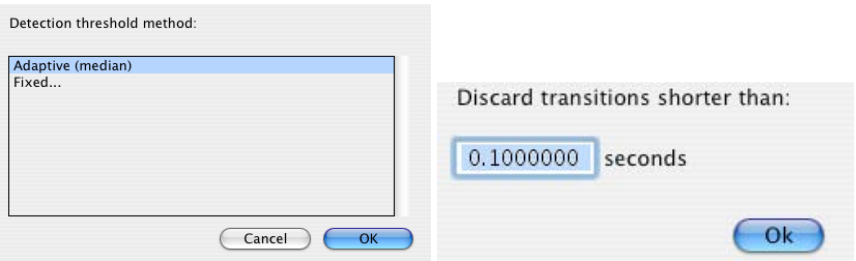
RMS EMG is computed using the Integrate transformation in a Root Mean Square Average over Samples configuration.

### *EMG Frequency & Power Analysis*

Several frequency domain techniques may be used for data reduction of EMG signals. The EMG Frequency & Power Analysis script extracts several measures derived from the power spectrum of an EMG signal. The EMG signal is split up into a fixed number of time periods; within each window, the power spectrum is computed using the Power Spectral Density transformation. For each time period, the following measures are extracted:

| *Name* | *Abbrev.* | *Description* | *Units* |
|---|---|---|---|
| Median Frequency | MedianF | Frequency at which 50% of the total power within the epoch is reached. | Hz |
| Mean Frequency | MeanF | Frequency at which the average power within the epoch is reached. | Hz |
| Peak Frequency | PeakF | Frequency at which the maximum power occurs during the epoch. | Hz |
| Mean Power | MeanP | The average power of the power spectrum within the epoch. (Units Note: V will be replaced with the voltage units in which the EMG was recorded) | $\frac{V^2}{Hz}$ |
| Total Power | TotalP | The sum of the power at all frequencies of the power spectrum within the epoch. (Units Note: V will be replaced with the voltage units in which the EMG was recorded) | $\frac{V^2}{Hz}$ |

### *Locate Muscle Activation*




When performing gait analysis, exercise physiology, or other research, identification of periods where the muscle is active can allow for correlation of external factors to muscle activity. Locate Muscle Activation attempts to identify various periods of muscle activity using statistical methods. The transformation requires a raw, unfiltered surface EMG channel. It is important that the muscle being examined is relaxed for the first

0.25 seconds of the recording to provide an estimate of the "background noise" during areas of muscle relaxation. This quarter-second period is used to estimate baseline parameters that affect the entire process.

This transformation implements a variation of the Hodges and Bui detection algorithm as described in:

P. W. Hodges and B. H. Bui, "A comparison of computer-based methods for determination of onset of muscle contraction using electromyography," *Electroenceph. Clin. Neurophysiol.*, vol. 101, pp. 511-519, 1996.

The variation implemented is a threshold-based algorithm roughly consisting of the following steps:

1. Determine mean value $\mu_0$ and resting standard deviation $\sigma_0$ of the first 0.25 seconds of the signal.
2. Construct a filtered ARV EMG signal, *z*.
3. Extract the variance of the signal with respect to the noise with the formula
$$g = \frac{z - \mu_0}{\sigma_0}$$
4. Using a threshold *h*, determine when the signal *g* lies below and above the threshold. Portions of time above the threshold are periods of muscle activity.
5. Discard any transitions across the threshold if they are shorter in duration than a user-specified time, *t*.

There are two methods of specifying the threshold *h*. An adaptive method examines the signal *g* and chooses the threshold to be the median of *g* over the entire waveform. Alternatively, the threshold can be specified manually. Using a manual threshold can be useful in adjusting the detection to better match specific EMG data. A suggested threshold is 2.5. By lowering the threshold, a larger quantity of data will be considered as muscle activity. By raising the threshold, a larger quantity of data will be considered to be noise.

The transition discard time *t* is specified in seconds. The default value of *t* is 0.1 seconds. If muscle activity is being inaccurately identified as inactivity for short periods within active times, try increasing the value of *t*. Do not set *t* to be greater than the smaller of either the shortest duration of a single muscle contraction or the shortest rest interval inbetween consecutive muscle contractions.

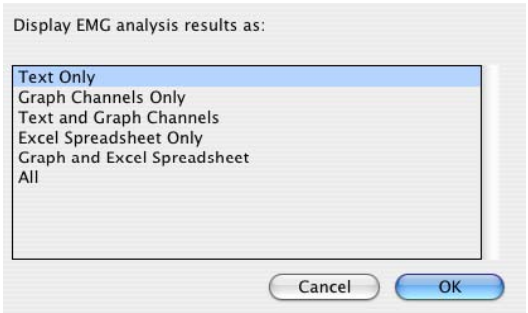
There are two outputs from the Locate Muscle Activation script.

- A new waveform, Muscle Active, will be added to the graph. The value of this wave will be zero when the muscle is at rest and one when the muscle is active. This wave can be used to quickly visually examine the record for periods of activity.
- Events are also generated on the raw EMG waveform. A Waveform Onset event is placed at each transition from inactive to active, and a corresponding Waveform End event is placed at each active-to-inactive transition. You can use these events in conjunction with the Cycle Detector to perform further data reduction based on muscle activity.

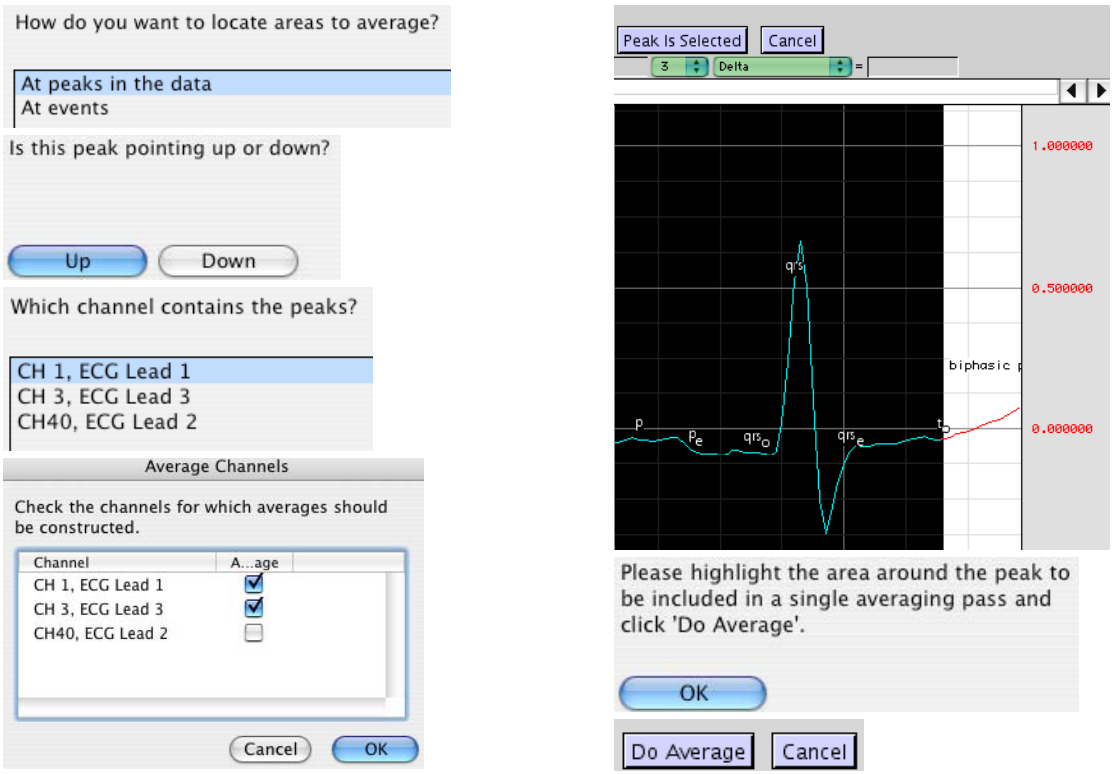
The detection of muscle activation onset and end from surface electrode EMG is an imprecise process. The output of this location should be visually examined for misidentification of activation periods that are too short, too long, overlapping, or missed.

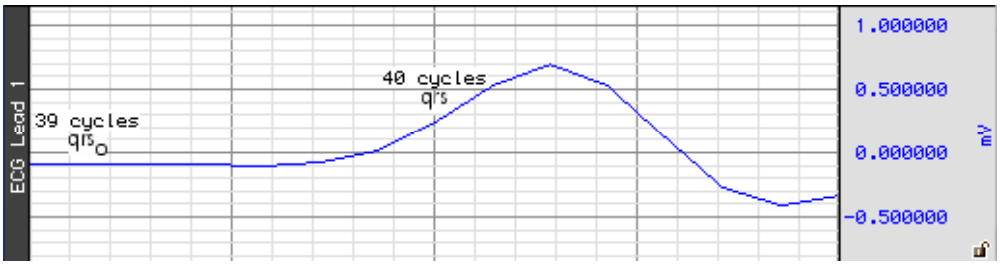
### *Preferences…*

The Preferences allow the type of output to be chosen for displaying results: text, graph channels, or Excel.



## Ensemble Average



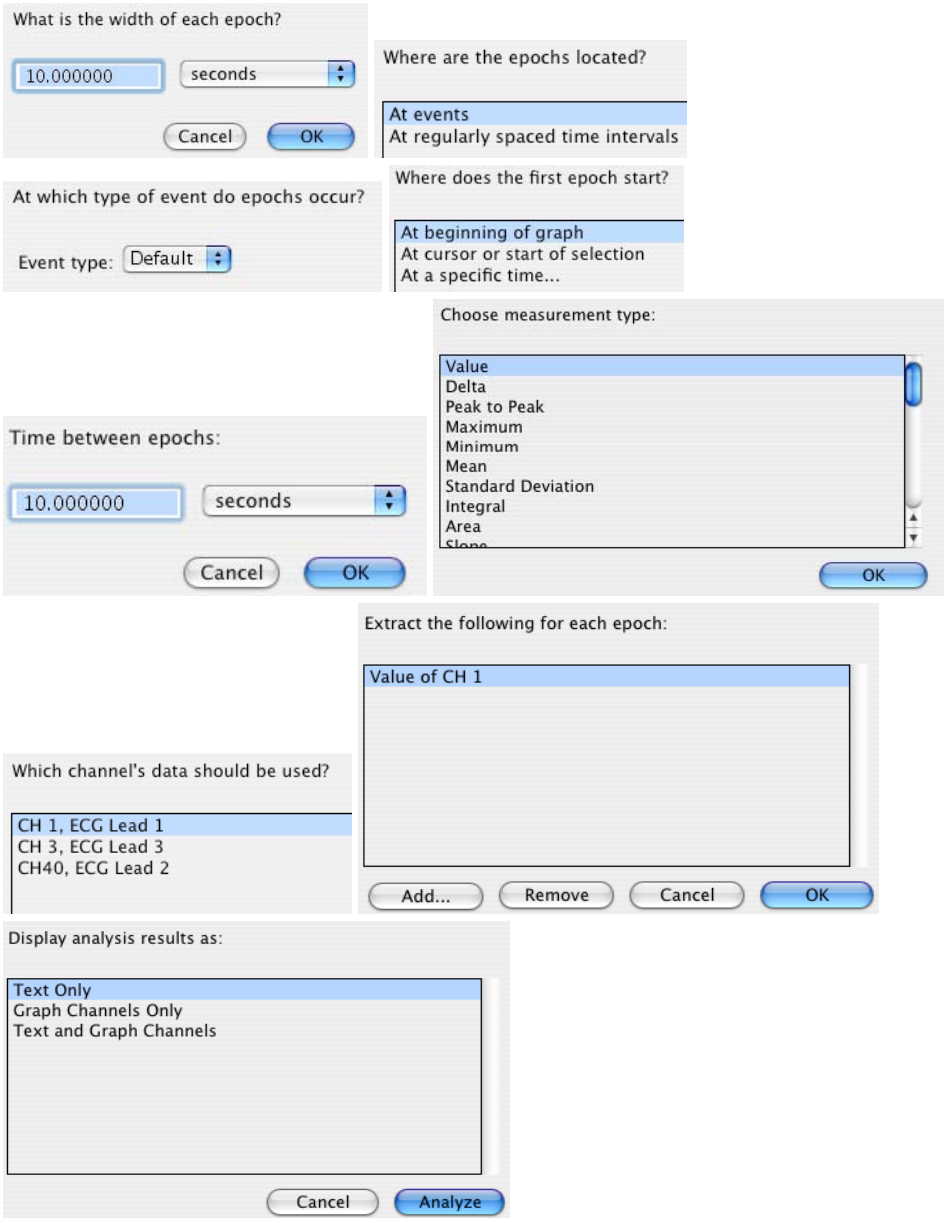




Ensemble Average assists in performing offline averaging. Offline averaging produces an average waveform from a number of cycles, also known as an *ensemble average*. Averages of multiple channels can be extracted simultaneously and be consolidated into a single graph window showing the results.

This option provides two methods for locating individual members of the ensemble.

- Data-driven peak detection with positive or negative peaks in the data. This method automatically derives appropriate threshold levels from a user-selected peak and is useful for constructing averages keyed to periodic signals with strong spikes, such as ECG.
- Place members of the ensemble surrounding events in the waveform. Events must be previously defined by the user, either manually or through another automated process. This method is useful for constructing averages keyed to any types of events in a graph.

## Epoch Analysis




Extracts basic measures from fixed-width time segments of data. A fixed-width time segment of data is known as an *epoch*. The location of these fixed-width intervals can either be keyed off of locations of events in the graph or tied to regular time intervals (e.g. occurring at a constant frequency). All of the standard Acq*Knowledge* measurements can be extracted on an epoch-by-epoch basis with the exception of calculation measurements.

Epoch-by-epoch measurement results can be viewed either as channels of data in the graph, a textual summary, or on an Excel spreadsheet; textual summaries include a final row with an overall average of each extracted measurement.

Times output by Epoch Analysis are always expressed in seconds; all other units correspond to the current preferred measurement unit settings accessible under Display > Preferences.

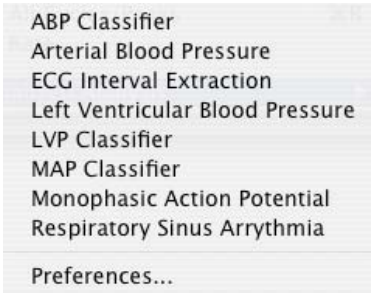
## Hemodynamic Analysis

Hemodynamics is the study of blood and circulation related data. This analysis package concerns itself with interpretation of ECG, blood pressure, and monophasic action potential data.

IMPORTANT: These routines are designed specifically for human subjects and may not function well, or at all, on animal subjects, particularly small animals.

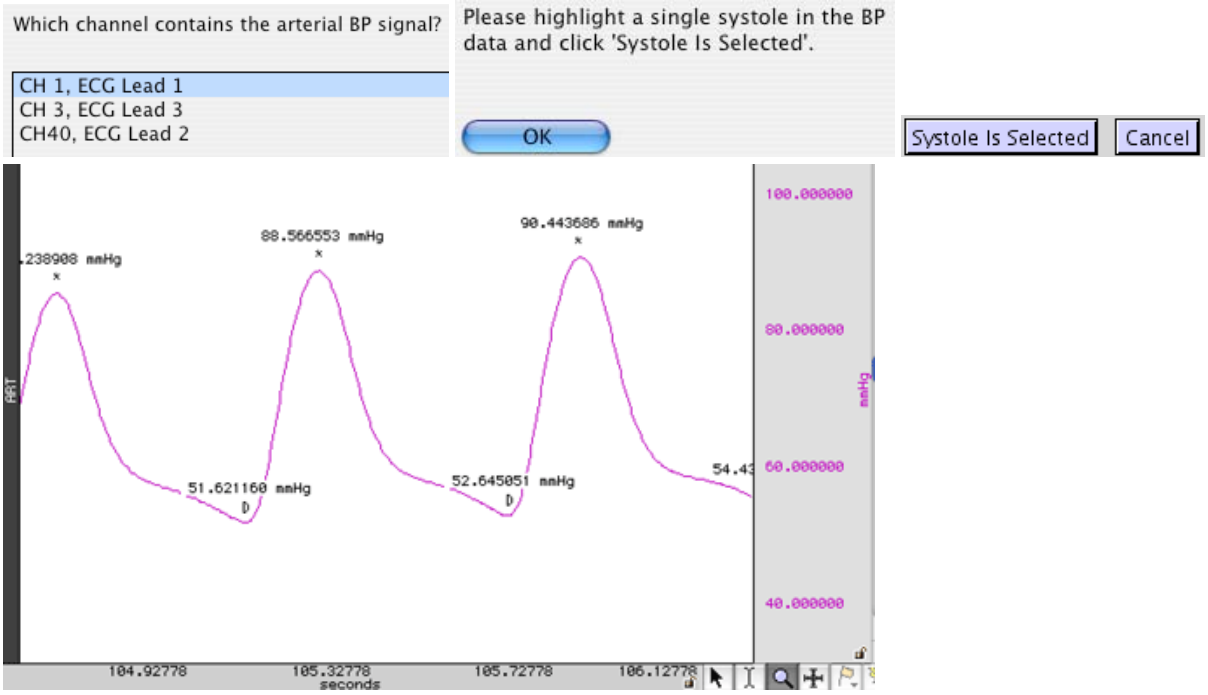
The Hemodynamic analysis package consists of:

a) ABP Classifier
b) Arterial Blood Pressure
c) ECG Interval Extraction
d) Left Ventricular Blood Pressure
e) LVP Classifier
f) MAP Classifier
g) Monophasic Action Potential
h) Respiratory Sinus Arrhythmia
i) Preferences



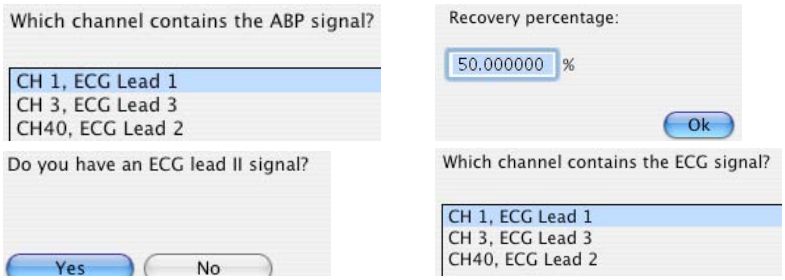

The time units reported by all of these transformations are in seconds unless otherwise noted.

### *ABP Classifier*




Places systolic and diastolic events at appropriate locations on a continuous arterial blood pressure signal recording using either invasive means or a continuous noninvasive pressure monitoring system. The ABP classifier functions directly on the pressure data and may fail for signals that exhibit strong noise characteristics or large baseline drifts. Pre-filtering the signal may improve classification accuracy.

### *Arterial Blood Pressure*




Extracts various cycle-by-cycle measures from arterial blood pressure (ABP) and ECG signals. It can function on an individual ABP signal or, when used in conjunction with an ECG Lead II signal, extract additional Q relative measurements.

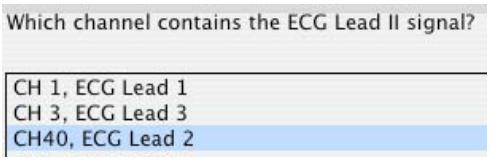
- If the ECG and ABP signals have not been classified when this analysis is performed, events for diastolic, systolic, and ECG boundaries will be inserted as necessary.
- If systolic, diastolic, and Q events are already present on the signals, however, they will be used.

On a cycle-by-cycle basis, the arterial blood pressure analysis transformation extracts the following measures:

| Name | Abbrev. | Description |
|---|---|---|
| Diastolic | - | Minimum pressure occurring during the cycle |
| Ejection time | ET | Time interval between the diastolic pressure and the minimum of dP/dt |
| Heart rate | HR | Heart rate in BPM as extracted from the diastolic-to-diastolic time interval for a given cycle |
| Maximum dP/dt | dP/dt max | Maximum amount of the change in the pressure during the cycle |
| Mean blood pressure | MBP | Mean blood pressure: $P_{diastolic} + \frac{P_{systolic} - P_{diastolic}}{3}$ |
| Minimum dP/dt | dP/dt min | Minimum amount of change in the pressure during the cycle |
| QA Interval | QA | Time interval between ECG Q wave and the diastolic pressure |
| Recovery interval | %REC | Time required for the pressure signal to decrease by a user specified percentage of the pulse height |
| Systolic | - | Maximum pressure occurring during the cycle |
| Time to peak pressure | TTPK | Time interval between the diastolic and the systolic pressures |

When textual output is used, the average of all of these measures will be output as the last row of the table.

### *ECG Interval Extraction*




Extracts cycle-by-cycle time and voltage measurements for various points and intervals between waveforms in the cycle on ECG Lead II signals. This interval extraction is based off of the waveform boundary locations with additional logic for defining explicit Q and S wave events. QRS peak events as output for boundary location are used as the R peak location.

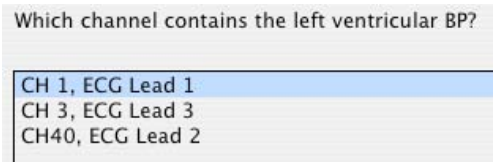
- If the ECG signal was not classified before running the interval extraction analysis, it will be classified automatically.

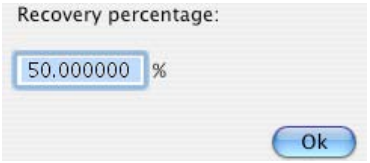
This analysis extracts the following cycle-by-cycle measures:

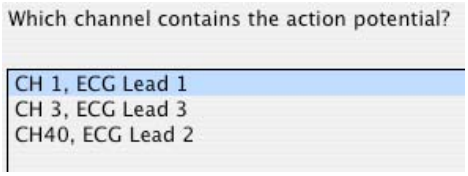
| *Name* | *Abbrev.* | *Description* |
|---|---|---|
| Corrected QT interval | QTC | QT interval divided by the square root of the RR interval |
| Heart rate | HR | RR interval expressed in BPM |
| P height | P-H | Amplitude at the peak of the P wave in a cycle |
| PRQ interval | PRQ | Time between the onset of the P wave to the Q wave |
| QRS width | QRS | Time between onset of the QRS complex and the end of the QRS complex. Equivalent to the time between onset of Q and end of S |
| QT interval | QT | Time between the beginning of the Q wave and the end of the T wave |
| R height | R-H | Amplitude of the R wave in a cycle |
| RR interval | RR-I | Time between consecutive R peaks in the waveform |
| ST interval | ST | Time between the S wave to the end of the T wave |

At the end of the text table output, the average of all of the cycles will be displayed. Additionally, both text and Excel output will indicate the number of cycles that did not have all three of the QRS, P, and T waves defined. These are cycles where the classifier missed a boundary and are listed as "Bad cycles," which may happen due to noise or other artifacts in the signal.

### *Left Ventricular Blood Pressure*




Extracts various cycle-by-cycle cardiac measures of left ventricular blood pressure data, optionally in conjunction with an ECG Lead II signal. Examines the LVP signal, ECG, and derivative of the LVP signal.

- If the LVP and ECG signals have not been classified before this analysis is executed, they will be classified automatically.
- Derivatives of the LVP signal can be pre-existing or can be constructed automatically.
- If an ECG signal is not included, only pressure related measures will be extracted.

The analysis outputs the following information on a cycle-by-cycle basis and the textual output cites the average of all of these cycle-by-cycle measurements:

| *Name* | *Abbrev.* | *Description* |
|---|---|---|

| *Name* | *Abbrev.* | *Description* |
|---|---|---|
| Contractility index | CI | maximum value of dP/dt during the cycle divided by the pressure at that time location |
| Developed pressure | DP | Interval between end diastolic pressure and systolic pressure |
| dP/dt Max | - | Maximum change in pressure over the cycle |
| dP/dt Min | - | Minimum change in pressure over the cycle |
| End diastolic pressure | LVDEP | the end diastolic pressure for the cycle. This is not necessarily the minimum pressure during the entire cycle.<br>LVDEP is located on the LVP signal using the method set in the preferences. |
| Minimum pressure | MIN | Absolute minimum pressure occurring during the entire cycle. This is not necessarily equivalent to the end diastolic pressure |
| QA Interval | QA | Time interval between the Q wave of the ECG and the end diastolic pressure |
| Rate | - | heart rate in BPM as extracted from the time interval between consecutive end diastolic pressure locations |
| Recovery time | %REC | Time it takes for dP/dt to increase from the minimum dP/dt location to a user specified percentage of that minmum value |
| Systolic pressure | SYS | Maximum pressure occurring during the entire cycle |
| Tau | - | the monoexponential time relaxation constant tau computed on a cycle by cycle basis.<br>See “Computation of Tau” on page 359 for specifics. |
| Tension time index | TTI | Integral of the pressure between end diastolic and the time of minimum dP/dt |

## Computation of Tau

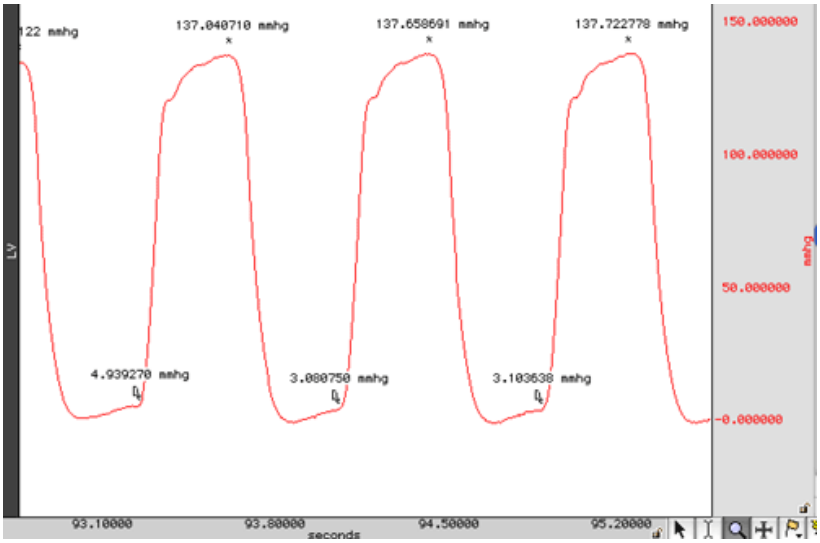
There are many different methodologies used to extract the time constant from LVP data. The time constant is extracted from a best fit parameter of a model to the trailing edge of LVP data on a cycle by cycle basis. This analysis uses a monoexponential model of zero asymptote for computing tau. The relaxation period is defined as the range of data between the time of minimum dP/dt in the cycle to the point where the LVP pressure signal drops below the previous LVDEP level. Within this range, the following model is fitted to the data using the simplex search method:

$$P_0 e^{\frac{t}{\tau}}$$

where $P_0$ is the value of the LVP signal at the time of dP/dt minimum and *t* is the time coordinate shifted such that *t* is 0 at the time of dP/dt minimum. The best fit value from this model is used as the value of the relaxation time constant.

### *LVP Classifier*

Operates on left ventricular blood pressure (LVP) data to define events at the systolic pressure and the left ventricular end diastolic pressure for each cycle.

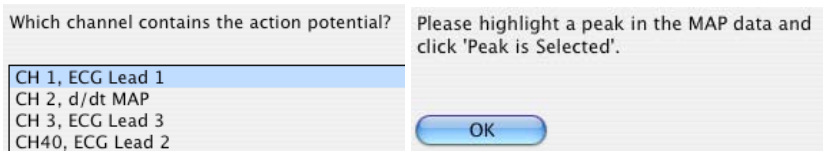



The location of these points is performed using filtered derivatives of the original LVP signal. Pre-filtering the signal (lowpass of 50Hz or less) or smoothing the signal before running the classifier may improve accuracy.

The LVP classifier locates LVDEP (left-ventricular end diastolic pressure) by examining the derivative of the pressure signal based upon the location method specified in Preferences:

- Adaptive threshold of 0 plus a percentage of the peak to peak change in pressure. The percentage is user-specified; the default is 1%. If the LVP signals do not have “flat” valleys, this percentage may need to be increased to fine-tune positioning of LVDEP.
- First zero crossings before contraction.

### *Monophasic Action Potential*




Performs classification of MAP data acquired from a human or animal subject and extracts various cycle by cycle intervals. Locates upstroke, maximum, 100% recovery, and user-specified recovery points on the action potential.

- Classification is performed using the action potential with its smoothed derivative; pre-filtering noise with lowpass filters may improve classification.
- If upstroke, maximum, and plateau events are already defined on the MAP signal, the classifier is not invoked and only recovery events are defined.

#### Plateau position

To better handle animal subjects and different potential morphologies, there are two methods for locating the plateau position in monophasic action potential data; use Preferences to set the method. Each method defines recovery percentage time locations depending on the signal between its maximum and the beginning of the plateau. The plateau is located by examining the derivative of the MAP immediately following its maximum value after an upstroke.

- The first method uses an adaptive threshold of zero plus a percentage of the peak to peak change in the derivative between the maximum and the first zero crossing after the maximum. If the signal remains above the upstroke voltage in this interval, a quick algorithm is used to locate 100% recovery and user-specified percentage levels. The default percentage is 0.1%, which will place the plateau position very close to the second zero crossing. This slight window around zero helps place plateau start events better for MAP data that has plateaus that continue increasing after their starting position.
- Searches for the second zero crossing after the maximum. If the signal drops below the voltage level of the upstroke in this interval, a different (slower) algorithm is used to ensure the recovery percentage is relative to the upstroke voltage and not the minimum occurring between the maximum and plateau.

The analysis outputs the following information on a cycle-by-cycle basis and the textual output cites the average of all of the cycle-by-cycle values:

| *Name* | *Abbrev* | *Description* |
|---|---|---|
| 100% recovery period | 100% REC | Time interval from the upstroke for the signal to recover back to the upstroke voltage level |
| dV/dt maximum | dV Max | Maximum change in voltage over the cycle |
| dV/dt minimum | dV Min | Minimum change in voltage over the cycle |
| End diastolic voltage | EDV | The value of the signal at the beginning of the upstroke |
| Max voltage | MAX | The maximum value of the signal over a single cycle |
| Minimum voltage | MIN | The minimum value of the signal over a single cycle. This may be less than the upstroke voltage depending on the morphology of the action potential |
| Plateau voltage | PLAT | The value of the signal at the start of the plateau after the completion of the upstroke |
| Rate | - | This is the heart rate in BPM as extracted from the time interval between consecutive upstrokes |
| Stroke amplitude | AMP | Voltage interval between the plateau and the upstroke voltage |
| User recovery period | %REC | Time interval from the upstroke for the signal to recover a specific percentage of the interval between the upstroke and the maximum voltage between the upstroke and the plateau |

### *MAP Classifier*

The classifier portion of Monophasic Action Potential only – defines upstroke, plateau, and percentage recovery events on MAP signals without performing the additional MAP data extraction.


The start of the plateau is located using either the second zero crossing of the derivative or a percentage of the cyclic peak-to-peak distance of the derivative. The plateau location method can be configured in Preferences.

### *Respiratory Sinus Arrhythmia*

Respiratory Sinus Arrhythmia is used to explore the connection between respiration and changes to heart rate. Variations in the heart rate can be directly correlated with vagal tone. The RSA index can be used to investigate changes in this connection during recording.

This RSA index is computed using the peak-valley method as outlined in:

> Grossman, P., van Beek, J., & Wientjes, C. (1990). A comparison of three quantification methods for estimation of respiratory sinus arrhythmia. *Psychophysiology*, 27, 702-714.

This method uses both a recorded ECG Lead II signal and a respiration signal. By using respiration information, this analysis method can provide breath-to-breath analysis that does not require parameter tweaking for individual subjects.

While designed for use with the RSP100C/TSD201 respiration module and transducer combination, it should be possible to use other estimates of respiration. The respiration signal is used to locate periods of inhalation and exhalation. Inhalation begins at valleys in the signal while expiration at peaks. Any respiration estimate that exhibits this morphology should be sufficient.

The RSA index outputted by this analysis is linearly scaled as per the recommendations in Grossman et. al. For comparison to other methods or studies using logarithmic scaling, Transform > Math Functions > Ln transformation can be used after analysis to convert results to logarithmic scaling.

RSA results are triggered from the respiration cycle. The RSA analysis outputs the following measures:

*Cycle* Index of the respiration cycle in the analysis.

*Time* Location of the start of the respiration cycle on the time axis.

*Min Rate* Minimum heart rate occurring during the inspiration window of the respiration cycle, expressed in milliseconds (IBI).

- If a respiration cycle is invalid, this measure will be set to 0.

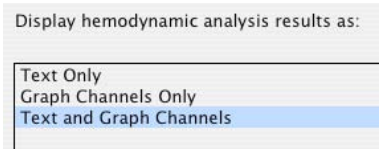
*Max rate* Maximum heart rate occurring during the expiration window of the respiration cycle, expressed in milliseconds (IBI).

- If a respiration cycle is invalid, this measure will be set to 0.

*RSA* RSA index for the respiratory cycle, expressed in milliseconds. This is the max rate minus the min rate. This is output in linear scaling. Conversion to logarithmic scaling must be performed manually, if desired.

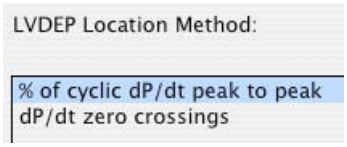
- If a respiration cycle is invalid, this measure will be set to 0.

### *Preferences*



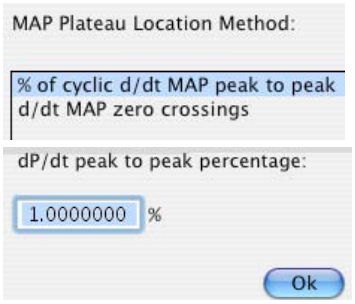

Display results as

Several of these transformations produce large amounts of cycle-by-cycle derived measures. Results can be displayed as a tab delimited table in the journal, as waveforms in the graph, as an Excel spreadsheet or various combinations. Results are displayed as text-only by default.




LVDEP location method – see page 360

- adaptive threshold of 0 plus a % of pk-pk change in pressure
- first zero crossings before contraction on the dP/dt signal

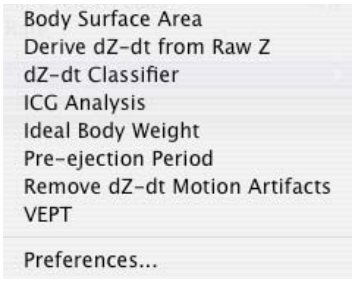



MAP Plateau location method – see page 361

- adaptive threshold of 0 plus a % of pk-pk change in the derivative between the max and the first 0 crossing after the max
- second zero crossing after the maximum

## Impedance Cardiography Analysis

The Impedance Cardiography analysis package assists in the analysis of cardiac output and other hemodynamic parameters using noninvasive bioimpedance monitoring techniques. It offers a variety of approaches for estimation of cardiac measures.




### *Body Surface Area*

Determines the body surface area estimation in square meters for a subject of a given height and weight, using the formula set in Preferences. Prompts require height and weight of the subject in the preferred units. It can be used to calculate body surface area independent of any of the other analysis routines, which may be useful for validation purposes or other derived calculations.

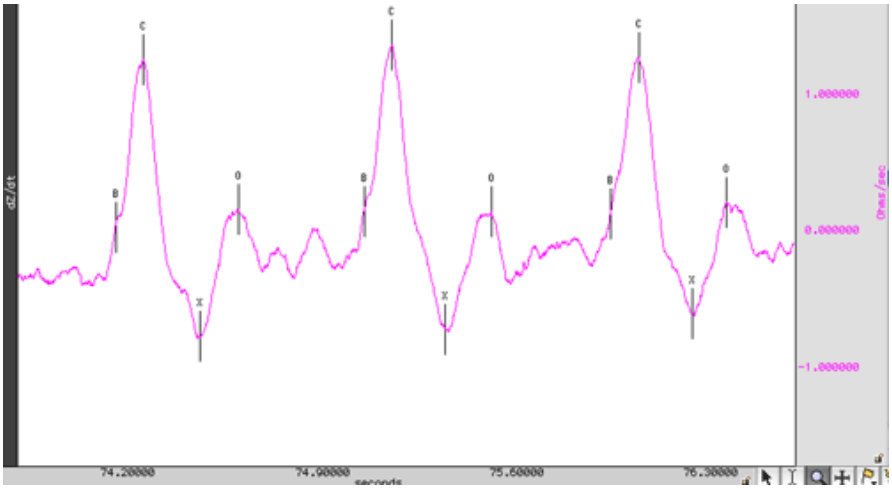
*Body Surface Area equation*

Use the Preferences option to select an algorithm for estimating body surface area of a subject and deriving stroke volumes from impedance data.

| *Method* | *Formula* |
|---|---|
| Boyd | $BSA = 0.0003207 \times Height(cm)^{0.3} \times Weight(g)^{0.7285 - 0.0188 \log(Weight(g))}$ |
| DuBois and DuBois | $BSA = 0.20247 \times Height(m)^{0.725} \times Weight(kg)^{0.425}$ |
| Gehan and George | $BSA = 0.0235 \times Height(cm)^{0.42246} \times Weight(kg)^{0.51456}$ |
| Haycock | $BSA = 0.024265 \times Height(cm)^{0.3964} \times Weight(kg)^{0.5378}$ |
| Mosteller | $BSA = \sqrt{\frac{Height(cm) \times Weight(kg)}{3600}}$ |

### *dZ/dt Classifier*

Places events at common inflection points on a dZ/dt waveform to derive other measures.




The classifier will attempt to locate the following points on the ICG signal:

- B point – opening of aortic valve (set location in Preferences)
- C point – Maximum left ventrical flow (set location in Preferences)
- X point – Closing of aortic valve (set location in Preferences)
- Y point – Closing of pulmonal valve
- O point – Widest opening of mitrial valve

The algorithm for locating these points on the ICG signal examines local minima and maxima in the dZ/dt signal as well as values of its second derivative. Filtering is applied to the second derivative signal to improve accuracy.

- Prefiltering the dZ/dt signal may improve accuracy slightly.

In a particular cardiac cycle, if there is not enough definitive change in the ICG signal to locate a particular point, the point will be omitted. This may most commonly occur with the Y point since its inflection between X and O is subtle and may be lost.

The location routine, as with impedance cardiography measurements in general, is sensitive to motion artifacts. It is intended to function on signals acquired from subjects at or near perfect rest. Swings in the dZ/dt signal may cause the classifier to fail. It is recommended that motion artifacts be removed before running the dZ/dt classifier or any other ICG analysis tools that may invoke the classifier on an ICG signal. If artifacts are present within the signal, the template matching cycle location method will exhibit better behavior than the fixed threshold method. The choice between these two methods can be made with the Preferences option of the analysis package.

*B-point Location*—Use Preferences to set the dZ/dt B-point location method.

There is no standard method generally accepted for programmatically locating B-points on an ICG waveform. The appropriate choice of B-point location method may depend on the data or on subjective preference. On average, all three methods will produce similar results for clean data. ICG Preferences has three options for B-point location:

- Second derivative classification – Given a C peak, it searches within a 150ms to 100ms time window before the C peak for the maximum of the second derivative of impedance ( $\ddot{Z}$ ). The B point is placed at this maximum.
- Third derivative classification – Given a C peak, it searches for the maximum value of the third derivative of impedance ( $\dddot{Z}$ ) within 300ms before the C peak. The B point is placed at this maximum.
- Cycle-by-cycle ‘Isoelectric’ crossings – Given a cycle defined by two C peaks, the mean of the dZ/dt signal is computed over the cycle. The B point is then placed at the closest time to the right C peak that is still underneath this baseline zero level.

*C-point Location*—Use Preferences to set the dZ/dt C-point location method.

In several of the ICG analysis scripts, the B, C, X, Y, and O points will need to be located on the dZ/dt waveform. The starting point of this process is locating individual cycles on the dZ/dt waveform to define the C points. Use Preferences to set the cycle location method:

- Template Matching – the user is expected to select a representative cycle of the dZ/dt waveform. The entire cycle should be selected (e.g. visually to approximate a C-C interval, a X-X interval, etc.). The entire dZ/dt signal is then correlated with that representative cycle, and individual cycles are picked out from locations of maximum correlation.
- Fixed Thresholding – the user is prompted to select one of the C peaks of the dZ/dt waveform. The voltage level of this peak is then used to compute an Ohms/sec thresholding level. Peak detection is then run on the dZ/dt waveform using that voltage level as the threshold.

  Since ICG is subject to many artifacts such as respiration components and motion artifacts, the default method used is template matching. For extremely clean ICG signals, however, fixed thresholding can be used effectively as well and will provide a quicker analysis.
- Adaptive template matching – the user is prompted to select a representative cycle of the dZ/dt waveform. This is used as a basis for an adaptive match to locate cycles. Adaptive template matching will adapt to changes in the dZ/dt waveform as conditions change within the experiment. Two parameters may be set. The window size is the number of ICG cycles to use for estimating the next template. Smaller values will track changes more quickly; larger values will reduce interference from artifact. The correlation threshold is the value above which a match is found. It refers to the normalized cross correlation of dZ/dt with the template and should be between 0 and 1. Values closer to 1 will require precise matches and skip artifacts. Values closer to 0 will use looser match constraints and may be required if the ICG is changing rapidly.

*X-point Location*

There are two methods that may be used to locate the X point of the ICG waveform at the closing of the aortic valve. The choice of appropriate X point location method is dependent on the electrode configuration that is used to acquire the ICG signals. In certain electrode configurations, the dZ/dt minimum may actually occur closer to A than to X, making the first (and default) option of searching for the first turning point a more reliable solution. You may want to acquire a phonocardiogram in conjunction with ICG to help determine which method will be more accurate at locating X.

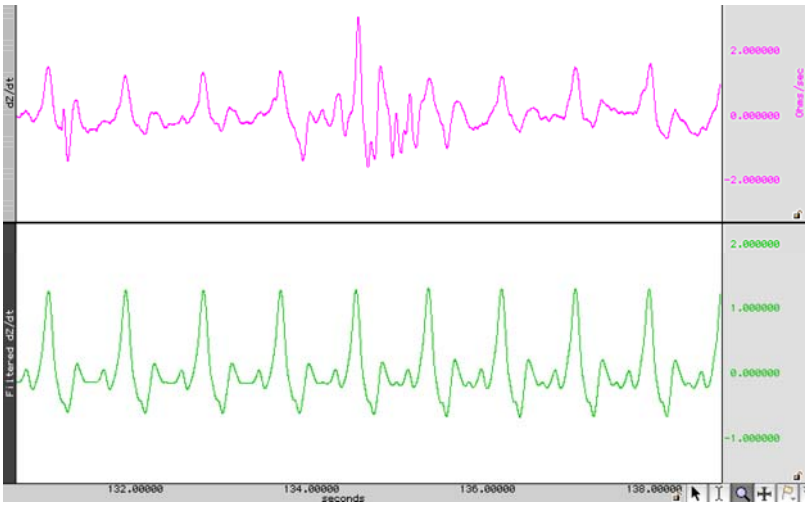
- Search for the first turning point in the dZ/dt signal that occurs after the C point location and place X at the first positive zero crossing in the second derivative of impedance ($d^2Z/dt^2$).This is the default X point location method.
- Locate the X point at the minimum value of dZ/dt over each cardiac cycle.

### *dZ/dt Derive from Raw Z*

This is a convenience utility for working with impedances recorded using the BIOPAC EBI100C amplifier or the raw impedance output of the BIOPAC NICO100C module. When computing derivatives from raw impedance signals from an EBI100C, this will apply appropriate filtering for a thoracic impedance signal and properly invert the derivative to match traditional dZ/dt presentation.

### *dZ/dt Remove Motion Artifacts*

Applies SFLC motion artifact removal to a dZ/dt signal. Uses cycle information from an ECG signal to construct a sinusoidal model of the ICG signal containing only components that are correlated to the heart rate.




IMPORTANT

Motion artifact removal will affect the amplitudes of the dZ/dt signal, so results derived from a motion filtered dZ/dt signal should be additionally verified for accuracy.

This tool performs the same type of filtering as the ICG Analysis and Pre-ejection Period tools when the Motion Filtering preference is enabled.

### *ICG Analysis*

Performs a full impedance cardiography analysis on data, extracting intervals and derived cardiac measures. The minimal set of signals required to run this analysis is an ECG Lead II signal and either a raw impedance signal or a dZ/dt signal.

- If a raw impedance signal is present from an EBI100C or NICO100C and no derivative has been constructed, the analysis will automatically construct the appropriate derivative and perform classification.
- If both a raw impedance and a dZ/dt signal are present, the baseline impedance will be derived on a cycle-by-cycle basis to improve the accuracy of the analysis.
- If no raw impedance signal was acquired, a default fixed baseline impedance can be used.
- If a NICO100C amplifier is used, it is recommended that both the raw impedance and dZ/dt signals be acquired to improve analysis accuracy.
- To automatically apply motion filtering to the dZ/dt signal, use Preferences to enable Motion Filtering (see page 372).

In addition to the minimal set of signals, it is also possible to use arterial blood pressure, central venous pressure, and pulmonary arterial pressure signals to improve the quality of the algorithm results. If any of these signals are not present, default fixed estimated values can be substituted for the mean pressures instead of deriving pressures on a cycle-by-cycle basis.

ICG Analysis may potentially perform classification of both the dZ/dt and the ECG Lead II signals. The various notes for understanding the limitations of these classifiers apply and should be understood to properly interpret failures in the analysis.

ICG Analysis will produce the following information on a cycle-by-cycle basis:

At the end of the textual table an average of all of the cycle-by-cycle values will be appended.

| *Name* | *Abbv.* | *Description* | *Units* | *Formula* |
|---|---|---|---|---|
| Acceleration index | ACI | Maximum blood acceleration | 1 / sec^2 | $\frac{\frac{d^2Z}{dt^2}_{\max}}{TFI}$ |
| Cardiac index | CI | Normalized cardiac output | m^2 / min | $\frac{CO}{BSA}$ |
| Cardiac output | CO | Volume of blood pumped each minute | l / min | $SV \times HR$ |
| Heart rate | HR | Heart rate in BPM as computed from the RR interval. | BPM | $\frac{60}{RR_i}$ |
| Left cardiac work | LCW | Work exerted by the left ventricle each minute | kg m | $(MAP - PAP) \times CO \times 0.0144$ |
| Left cardiac work index | LCWI | Normalized left cardiac work | kg m / m^2 | $(MAP - PAP) \times CI \times 0.0144$ |
| Left ventricular ejection time | LVET | Interval between B and X.<br>Time interval between aortic valve open and close. | sec | *Not applicable* |
| Mean blood pressure | MBP | Mean blood pressure as measured on the arterial blood pressure signal, or fixed estimate if no ABP signal is present. | mmHg | $P_{diastolic} + \frac{P_{systolic} - P_{diastolic}}{3}$ |
| Mean central venous pressure | CVP | Mean central venous pressure over cycle, or default value if no CVP signal is present. | mmHg | *Not applicable* |
| Mean pulmonary arterial pressure | PAP | Mean value of the pulmonary arterial pressure of a cycle, or default value if no PAP signal is present. | mmHg | *Not applicable* |
| Pre-ejection period | PEP | Interval between the Q wave of the ECG and the B point of the ICG.<br>Time interval between systole and aortic valve open. | sec | *Not applicable* |
| RR interval | RR-i | Interval between R peaks in the waveform. | sec | *Not applicable* |
| Stroke index | SI | Normalized stroke volume | (ml / beat)/ m^2 | $\frac{SV}{BSA}$ |

| *Name* | *Abbv.* | *Description* | *Units* | *Formula* |
|---|---|---|---|---|
| Stroke volume | SV | Volume of blood pumped by left ventricle in a single beat | ml / beat | Set equation in Preferences:<br>Kubicek—Estimates SV from the derivative of the impedance signal and blood resistivity:<br>$SV = \rho \times \frac{L^2}{Z_0^2} \times \frac{dZ}{dt}_{\max} \times LVET$<br>*Note* $\frac{dZ}{dt}_{\max}$ may be either the absolute maximum or the BC delta in amplitude, as set in Preferences.<br>▪ Sramek—Estimates SV from the derivative of the impedance signal and the estimated volume of electrically participating fluid (VEPT):<br>$SV = \frac{VEPT}{Z_0} \times \frac{dZ}{dt}_{\max} \times LVET$<br>o In the ICG analysis routines, VEPT is estimated using a truncated cone model.<br>$VEPT = \frac{(0.17H)^3}{4.25}$<br>▪ Sramek-Bernstein—Estimates SV from the volume of electrially participating tissue scaled according to body habitus. The SV equation is:<br>$SV = \frac{\delta(VEPT)}{Z_0} \times \frac{dZ}{dt}_{\max} \times LVET$<br>*where*<br>$\delta(VEPT) = \frac{weight_{actual}}{weight_{ideal}} \times \frac{(0.17H)^3}{4.25}$<br>Ideal body weight is computed using the method set in the Preferences. To best match the original Sramek-Bernstein equation, use the Met Life Tables ideal body weight method. |
| Systemic vascular resistance | SVR | Afterload; arterial flow resistance | dynes sec / cm^5 | $80 \times \frac{MAP - CVP}{CO}$ |
| Systemic vascular resistance index | SVRI | Normalized afterload | dynes sec m^2 / cm^5 | $80 \times \frac{MAP - CVP}{CI}$ |
| Systolic time ratio | STR | Ratio between electrical and mechanical systole | none | $\frac{PEP}{LVET}$ |
| Thoracic fluid content | TFC | Electrical conductivity of the chest cavity | 1 / Ohms | $\frac{1}{TFI}$ |

| *Name* | *Abbv.* | *Description* | *Units* | *Formula* |
|---|---|---|---|---|
| Thoracic fluid index | TFI | Mean value of the raw impedance over the cycle, or fixed baseline value if no raw impedance signal is present. | Ohms | *Not applicable* |
| Velocity index | VI | Maximum velocity of blood flow in the aorta. | 1 / sec | $\frac{\frac{dZ}{dt}_{max}}{TFI}$ Note $\frac{dZ}{dt}_{max}$ may be either the absolute maximum or the BC delta in amplitude, as set in Preferences. |

### *Ideal Body Weight*

Body Weight is derived from a person's height, gender, and (for the Met Life method) frame size. It describes the ideal weight based upon various estimates. Ideal body weight is subject to much interpretation, so a number of methods are provided. Ideal Body Weight results are always expressed in kilograms.

Use Preferences to set the Ideal Body Weight computation method; the selected method is also used for ICG Analysis.

| *Method* | *Formula* |
|---|---|
| Devine | *Men* 50 kg + 2.3 kg per inch over 5 feet<br>*Women* 45.5 kg + 2.3 kg per inch over 5 feet |
| Metropolitan Life Tables | The weight is taken from the standard Metropolitan Life tables, which are based on gender, height, and frame size. The Metropolitan Life tables specify weight ranges; the ideal body weight is computed as the average of the endpoints of each weight range. Ideal weights are based on height with shoes on and are only defined for heights between<br>*Men* 5' 2"and 6' 4"<br>*Women* 4' 10"and 6' 0" |
| Miller | *Men* 56.2 kg + 1.41 kg per inch over 5 feet<br>*Women* 53.1 kg + 1.36 kg per inch over 5 feet |
| Robinson | *Men* 52 kg + 1.9 kg per inch over 5 feet<br>*Women* 49 kg + 1.7 kg per inch over 5 feet |

### *PEP Pre-ejection Period*

The pre-ejection period is the time interval between the electromechanical systole and the onset of ejection of blood from the left ventricle of the heart. This can be derived from standard ECG data and ICG data as the interval between the Q point on the ECG and the B point on the ICG. The Pre-ejection Period analysis tool helps extract PEP measurements from ECG Lead II and ICG data. PEP can also be computed using the full ICG Analysis tool on page 367.

To use Pre-ejection Period analysis, both an ECG Lead II and an ICG (dZ/dt) signal must be present. If either of these signals requires classification, the analysis will run the appropriate classifier to define the relevant events on the signals. To automatically apply motion filtering to dZ/dt, use Preferences to enable Motion Filtering (see page 372).

PEP analysis will output the following information on a cycle-by-cycle basis and the final line of the textual output will be the average of all of the cycle measurements. All time unit output is in seconds unless otherwise noted.

| *Name* | *Abbrev.* | *Description* |
|---|---|---|
| Heart rate | BPM | The heart rate for the cycle as indicated in BPM. Derived from the RR interval. |
| Pre-ejection period | PEP | Interval between the Q wave of the ECG and the B point on the ICG for the cardiac cycle.<br>If the PEP cannot be computed for a particular cycle, it will have the value "----" in the textual output or 0 in the graphical output. |
| RR interval | RR-i | Time interval between R peaks of a single cycle of cardiac data. |

### *VEPT*

Uses the truncated cone method to compute the volume of electrically participating tissue (VEPT) in cubic centimeters of a subject. You will be prompted to enter the height of the subject in the preferred units. It can be used to calculate VEPT independent of any of the other analysis routines, which may be useful for validation purposes or other derived calculations.

### *Preferences*

The following Preferences are available in the following order. You must click through all Preferences to adjust a particular Preference.

Body surface area equation – see page 364

- Boyd; DuBois and DuBois; Gehan and George; Haycock; or Mosteller

Ideal Body Weight method– see page 370

Stroke volume equation – see page 369

- Kubicek, or Sramek, or Sramek-Bernstein

Output display format

- Textual tables in the journal
- Channels of data inserted into the graph.

Unit system for inputting body measurements

- English system: body height in feet and inches, distance between measuring electrodes in inches, and body weight in pounds
- Metric system: body height in meters and centimeters, distance between measuring electrodes in centimeters, and body weight in kilograms.

dZ/dt B-point location – see page 365

dZ/dt C-point location –see page 366

dZ/dt X-point location – see page 366

dZ/dt Motion Filtering Automatic Filtering Preference

The Pre-ejection Period and ICG Analysis transformations have the ability to optionally apply motion filtering automatically to the dZ/dt signal. Motion filtering is performed using an SFLC keyed to the R waves of an ECG signal. The SFLC filtering approach is similar to performing cycle-by-cycle averaging of the dZ/dt signal. This motion filtering approach may cause errors to be introduced in derived calculations, so any results with motion filtering turned on should be validated additionally.

*Filter Magnitude Preference* – relaxed, aggressive, and custom.

- "Relaxed" uses a SFLC step size of .001. This allows the filter to adapt moderately quickly to changes in the dZ/dt signal.
- "Aggressive" uses a SFLC setting of .0001. The filter will adapt much less quickly to changes in the ICG signal, allowing better filtering out of motion artifacts at the expense of a lessened response to changes in underlying ICG morphology.
- "Custom" allows for an arbitrary SFLC step size. The step size must be greater then zero and much less than 1 for the filter to converge.

dZ/dt Max method – Baseline drift in ICG signals can introduce drift artifacts into stroke volume, cardiac output, and other measures that are sensitive to changes in dZ/dt max. The Preferences offer two settings. "Max dZ/dt in cardiac cycle" will extract the maximum amplitude of dZ/dt as the max value. This is the traditional way of measuring dZ/dt max. A second estimate option, "change in voltage from B to C" will take the amplitude delta between B and C as the estimate of dZ/dt max. This will produce different stroke volume results, but is useful for removing motion artifact and improving consistency.

## Magnetic Resonance Imaging


Magnetic resonance imaging, or MRI, is often used to study the brain and other organs in the body. As access increases to MRI machines, researchers are beginning to combine MRI with traditional physiological signal recording. The strong magnetic fields used by MRI equipment can cause profound artifacts in physiological recordings, which can make the analysis of physiological recordings acquired in an MRI difficult. Some artifacts are external interference while other artifacts can be caused by currents being induced in electrode leads or even in the body itself.

### Artifact Location and Trigger Signals

Most of the MRI analysis options require information to identify the positions of various artifacts. Event positions can be used or a "trigger signal" waveform in the graph can be used to identify periods when the MRI is active. Some MRI machines have a TTL output that is synchronized with periods where the MRI is on.

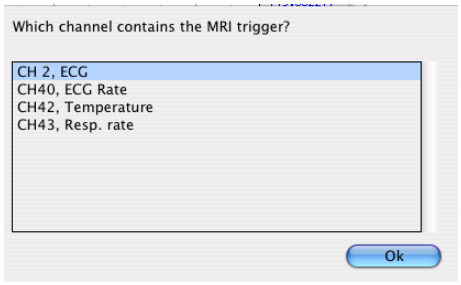
- Whenever possible, this trigger signal should be acquired with the MP unit along with the physiological data.

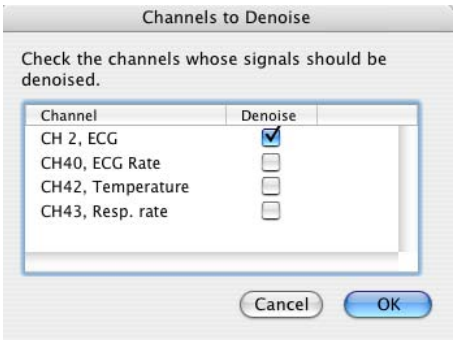
Trigger detection off of an MRI trigger signal waveform is performed using fixed level thresholding on the waveform data. The threshold level is set to be the minimum value of the entire trigger signal plus $1/10^{th}$ of the peak-to-peak distance of the trigger signal. The threshold is kept data dependent to allow for artificial trigger signals to be derived from data if the MRI unit does not provide its own. The trigger signal may be acquired on either an analog or digital channel.

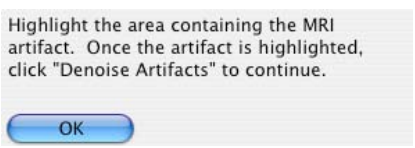
Event driven artifact location can be useful when trigger signals are not available from the MRI or are not recorded. A cycle detector analysis can be used to place events at the onset of each artifact, or these events may be placed manually. Event based detection is also useful for applying the procedures for artifacts that are not directly related to the MRI trigger signal, such as for removing the cardiac interference from EEG data caused by the magnetic field of the MRI machine.

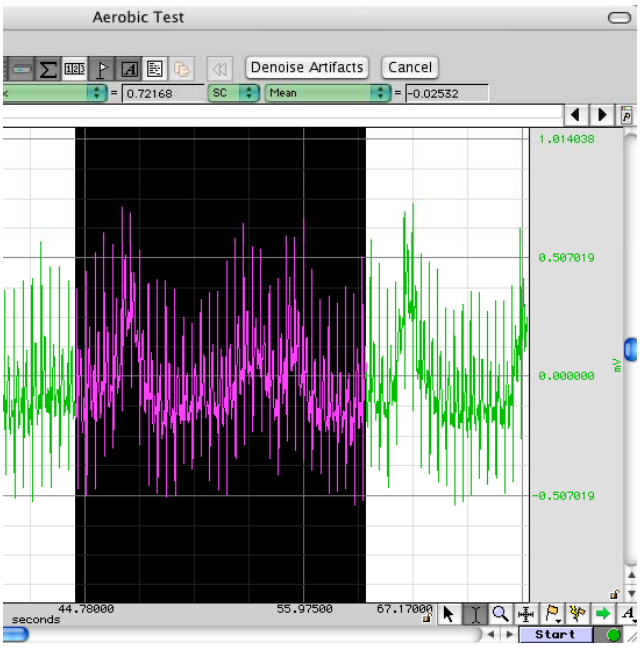
### *Artifact Frequency Removal*

MRI > Artifact Frequency Removal




Two large sources of interference in MRI recordings are the current induced by the MRI magnetic field and the RF pulses used for triggering molecule alignment. While the overlap of this interference may be difficult to separate in the time domain, the MRI interference may have a distinctive signature in the frequency domain.

Artifact Frequency Removal is a frequency domain adaptation of the ensemble projection removal of the Artifact Projection Removal transformation. It attempts to cancel out MRI artifact by removing the frequencies most strongly associated with the MRI signal.

For each channel of data to be denoised, either the MRI trigger signal or event positions are used to locate periods of MRI activity for constructing an ensemble average. The FFT of this ensemble average is computed, and the magnitude of the average FFT is set as the reference. Cyclic mean removal is applied to each period of artifact to compensate for baseline drift or signals with expected DC offset. A second pass is then made through the data. For each individual artifact, the FFT of that artifact is computed and the projection of that FFT onto the average FFT is removed. After projection removal, negative Fourier components are discarded and a time-domain signal is reconstructed using the inverse Fourier transform. This reconstructed, filtered signal is used to replace the MRI artifact in the original data.

Application of projection removal in the frequency domain has similar limitations to applying it in the time domain, that is, it assumes that the MRI interference is stationary (which is not necessarily the case). Variations in the MRI interference may cause this method to fail.

*IMPORTANT* Artifact Frequency Removal requires an MRI triggering signal or artifact onset events to locate artifact positions.

### *Artifact Projection Removal*

Artifact Projection Removal attempts to remove the noise components from the artifacts within a signal. An ensemble average is made for each period of MRI artifact in a channel. Cyclic mean removal is applied to each period of artifact to compensate for baseline drift or signals with expected DC offset. As the artifacts are averaged together, the actual interference with the physiological signal caused by the MRI should become the dominant feature if a sufficient number of artifacts are present. A second pass is made through the artifacts to remove this average MRI artifact from each individual period.

The average artifact is removed using the Remove Projection transformation. This performs a vector projection of the signal onto the averaged artifact estimation and subtracts this projection. This is an improvement over straight subtraction of the average artifact as vector projection can compensate for changes to amplitude that may occur over time.

Artifact projection removal cannot compensate for MRI interference that varies in frequency due to changes in orientation of electrode leads within the MRI or other factors that may alter the MRI artifact.

Artifact projection removal is an adaptation of a denoising technique described in:

M. Samonas, M. Petrou and A. Ioannides, "Identification and Elimination of Cardiac Contribution in Single-Trial Magnetoencephalographic Signals," *IEEE Trans. Biomed. Eng.*, vol. 44, no. 5, pp. 386-393, 1997.

*IMPORTANT* Artifact Projection Removal requires an MRI triggering signal or artifact onset events to locate artifact positions.

### *Median Filter Artifact Removal*

Median Filter Artifact Removal provides a basic artifact removal suitable for slow moving signals such as respiration, GSR, or temperature. It performs a windowed median transformation on the source channel with a window width of $1/10^{th}$ of the acquisition sampling rate.

This median filtering approach is explained in the BIOPAC MRI application note AH223.

Median Filter Artifact Removal does not require an MRI triggering signal.

### *Signal Blanking*

MRI artifact can grossly distort low level physiological signals, and this distortion can be several orders of magnitude larger than the signal of interest. A common practice for analyzing the physiological data is to discard the MRI artifacts and only examine the portions of the signal in between the MRI artifacts. One approach for this is outlined in BIOPAC MRI application note AH223.

Signal Blanking provides an alternate approach for discarding MRI artifacts from the signal. Using the MRI triggering signal or artifact event locations, this analysis option will locate the periods of MRI activity and "blank" the physiological signal during this period.

Two types of "blanking" can be performed:

- Set value to zero – The waveform is set to zero during each artifact.
    - For integrated measures, zeroing the signal may be preferable as it will have no effect on the running sum.
- Connect endpoints – For each artifact, a selection is made and the values within the interval are replaced with a line connecting the signal value before the MRI artifact to the signal value at the end of the interval.
    - For statistical measures or DC coupled signals, connect endpoint (linear interpolation within the interval) may be preferable to avoid causing the output to trend towards zero.

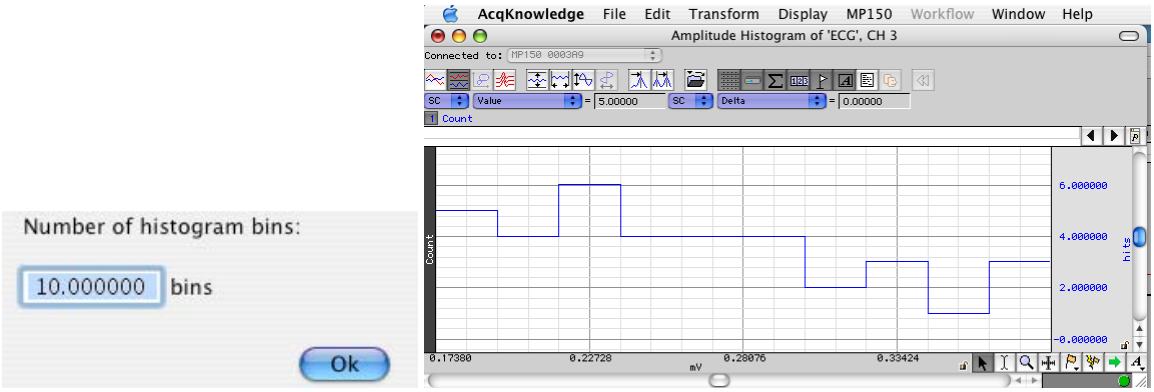
*IMPORTANT* Signal Blanking requires an MRI triggering signal or artifact onset events to locate artifact positions.

## Neurophysiology

The Neurophysiology analysis package assists in the analysis of spikes within extracellular microelectrode recordings, such as those recorded using an MCE100C module. All of these analysis options require a continuous recorded single channel of microelectrode data.

- A *spike* is a deviation from the baseline caused by a neuron action potential. Frequently extracellular spikes will resemble exponentials. The point of maximum value of the spike will be used to locate neuron firing.
- A spike *episode* consists of a fixed time window around a spike that aims to capture the underlying neuron firing time. The episode consists both of the rise time (the time taken to reach maximum) and the relaxation period around the spike.

### *Amplitude Histograms*

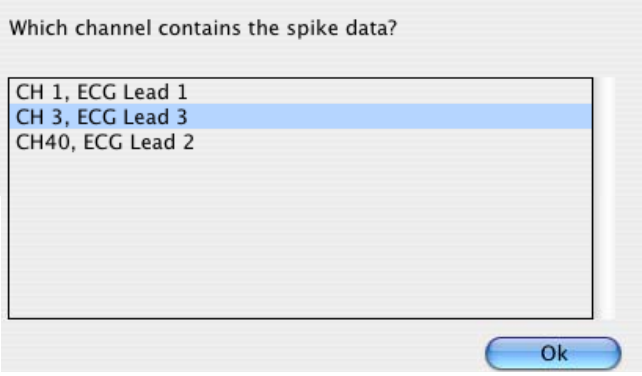



*IMPORTANT* To run this analysis option, the signal must first be transformed by the Locate Spike Episodes option or the Classify Spikes option.

Amplitude histograms show the population density of the maximum amplitude of neuron firing events. They may be used to interpret changes in neuron firing due to drug response or as rough indicators of the approximate number of classes of action potentials in a signal. Amplitude histograms can be generated on classified or unclassified signals.

- On classified signals, an overall amplitude histogram will be created for all of the spikes in addition to a single amplitude histogram per class (reflecting only the episodes of that class.
- On unclassified signals, a single amplitude histogram will be created from the maximum voltage within all of the spike episodes.

### *Average Action Potentials*

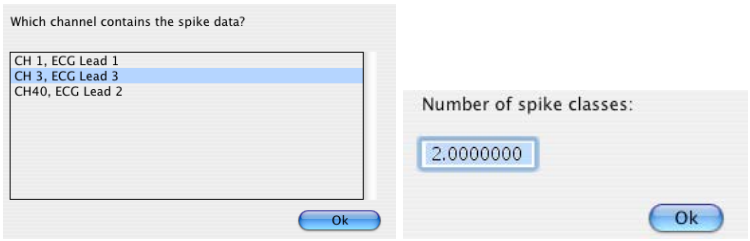



*IMPORTANT* To run this analysis option, the signal must first be transformed by the Locate Spike Episodes option or the Classify Spikes option.

After a classification has been completed for a spike signal, to assign spike episodes to different groups, users may wish to view the average shape of the waveforms of each class. Examining the shape of the different classes provides visual feedback as to the efficiency of the clustering, can allow for identification of certain classes as noise or artifacts, and helps to determine if the identified classes are indeed unique. Average Action Potentials can be generated on classified or unclassified signals.

- On classified signals, the resulting ensemble averages will have multiple channels.
  - The first channel will be the overall average of all of the spike episodes.
  - The remaining channels show the average of the members of each individual spike class.
- On unclassified signals, a graph will be produced with a single channel showing the average of all of the spike episodes.

### *Classify Spikes*




*IMPORTANT* If cluster events from a previous spike classification are already defined on the recorded waveform, this option will erase them and replace them with the new classification of the potentials.

This analysis option will automatically classify action potentials in microelectrode data and divide them into different spike classes.

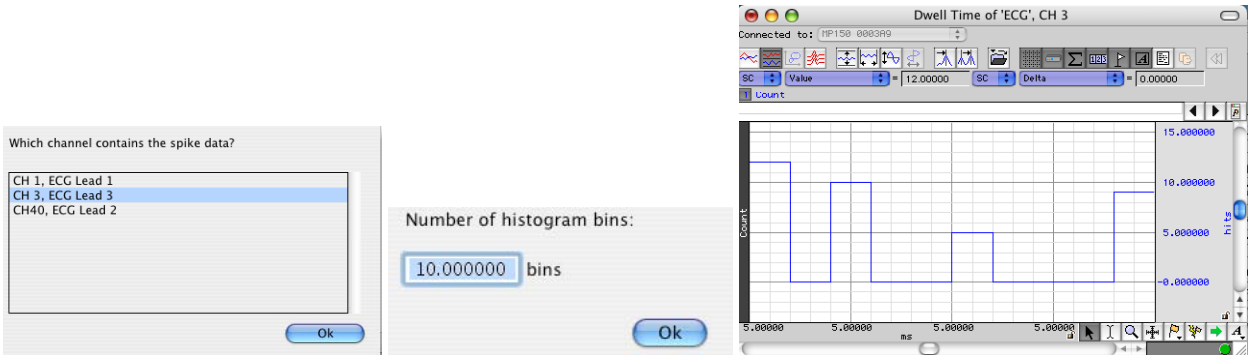
A single-feature k-means clustering classifier is used, and the entire data set is used for the clustering portion of the algorithm. The determining feature is the Sum criteria— that is, the sum of all of the data points within the waveform segment; this was one of the first features used in early action potential classifiers.

If the Locate Spike Episodes option has not been used to find spikes before this option was selected, the Locate Spike Episodes option will be automatically performed prior to the clustering.

The analysis routine will ask for a number of spike classes and then use k-means clustering to group each spike episode into a class. The clustering may not produce meaningful classes, so results should be examined for accuracy.

This style of classifier is for rudimentary spike analysis. For more advanced classification techniques, use the clustering algorithm in the peak detector.

### *Dwell Time Histograms*




*IMPORTANT* To run this analysis option, the signal must first be transformed by the Locate Spike Episodes option or the Classify Spikes option.

A dwell time histogram shows the population density of the duration of a neuron firing event. Dwell times can be approximated for an action potential by measuring the absolute value of the time interval between their maximum and minimum voltage levels reached during the firing of the neuron. After the minimum value in the firing recording has occurred, the neuron will be returning to its resting state, so the time difference is a good approximation for the firing duration. The dwell time histogram plots this time difference versus number of action potentials that have similar time differences. Examining varieties in dwell times can help to illustrate drug responses or to perform rudimentary classification of action potentials.

Dwell times will be defined as the time difference between the positions of the maximum signal value and minimum signal value within a spike episode. Since dwell time histograms can be used for classification purposes, they can be run on classified or unclassified microelectrode signals.

- On classified signals, an overall dwell time histogram will be constructed for all of the spikes in addition to a single histogram per class, showing times of only the spikes in that class.
- On unclassified signals, a single dwell time histogram will be created for all of the spikes. When run on a classified signal.

### *Find Overlapping Spike Episodes*

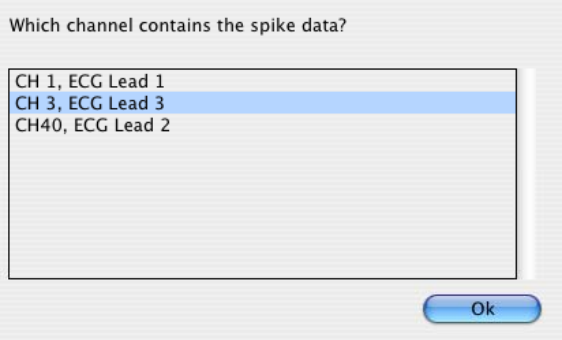
*IMPORTANT* To run this analysis option, the signal must first be transformed by the Locate Spike Episodes option or the Classify Spikes option.

In many extracellular recordings, it is frequent for there to be more than one neuron firing in response to the same stimulus. This can result in overlapping spike episodes when both neurons fire in close succession. Some types of analysis and spike classification are not able to produce meaningful results if too many overlapping episodes occur. "Find Overlapping Spike Episodes" can be used to locate overlapping episodes. After the spikes have been located in a signal, this option can be used to iterate only to those that are overlapping.

"Next Overlap" and "Cancel" buttons are available in the toolbar of the graph window to allow for iteration through the episodes.

*Note* This option is "view only." Overlapping episodes are not affected by the analysis and will need to be manually removed manually to delete them from the file.

### *Generate Spike Trains*




*IMPORTANT* To run this analysis option, the signal must first be transformed by the Locate Spike Episodes option or the Classify Spikes option.

Spike trains are good visual indicators of when action potentials are firing and are good synchronization waves for further analysis and data reduction. A spike train is a channel in a graph whose value is 0 if no spike is firing and 1 if a spike is firing.

Spike train generation will operate only on signals whose action potentials have already been classified.

A single spike train will be generated as a channel in the graph for each class of action potential in the signal.

If text output is enabled, the spike trains will be pasted as tables in the journal with one table per spike class.

If spreadsheet output is enabled, the tables will be placed side by side so index 1 of the tables lines up for each action potential.

### *Locate Spike Episodes*

Neurophysiology > Locate Spike Episodes


This option provides the basic spike detection for a microelectrode signal. Spike detection is performed using the following steps:

1. Obtain mean value of the entire signal.
2. Obtain standard deviation of the entire signal.
3. Detect spikes where the signal rises above a fixed threshold determined by adding a multiple of the standard deviation to the mean.
4. Position the episode around the threshold crossings according to the width and offset entered previously.

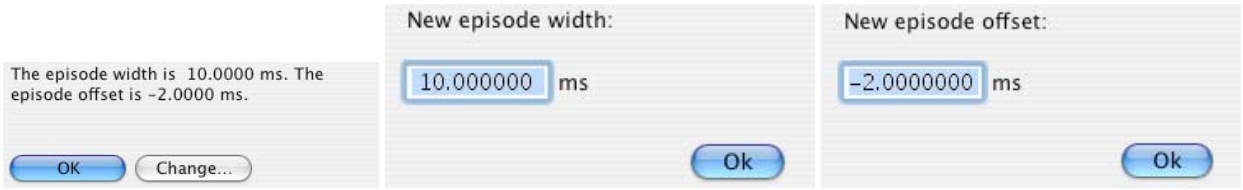
A "Spike Episode Begin" event will be placed at the start of each spike episode and will be located offset milliseconds away from the threshold crossing. A "Spike Episode End" event will be placed at the end of each episode.

If text output is enabled, a table of the start time of each episode will be placed in the graph's journal.

If spreadsheet output is enabled, a new spreadsheet will be created with the start time of each episode.

Spike episodes may also be located manually by using the Cycle Detector to define "spike episode begin" and "spike episode end" events in the graph.

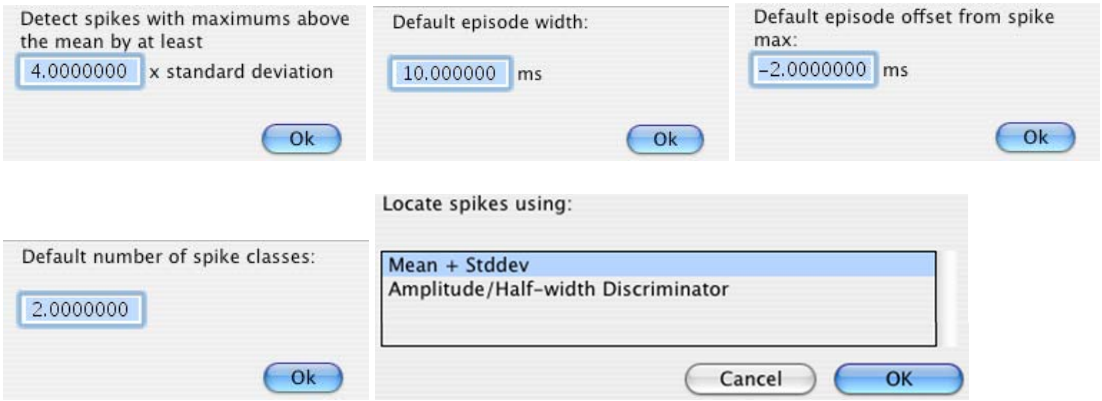
### *Set Episode Width & Offset*




The first time spike detection is performed on a graph, the episode width and offset need to be entered. This width and offset is remembered and is used for all future spike detections in the graph performed by "Locate Spike Episodes" and other transformations. The width and offset that are entered are retained even if the file is saved and reopened.

Use this option to view or change the current width and offset.

### *Preferences*

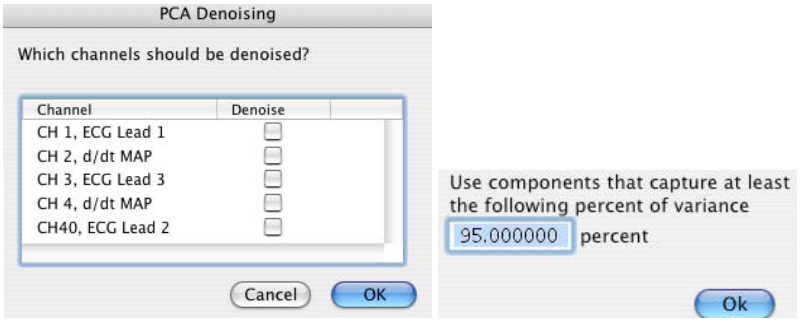



| *Preference* | *Description* | *Default Setting* |
|---|---|---|
| Output type | Determines whether analysis results will be displayed as graph channels, textual tables in the journal, or Excel spreadsheets. Not all of the output types are applicable for each Neurophysiology analysis option. | Text output to journal only. |

| *Preference* | *Description* | *Default Setting* |
|---|---|---|
| Mac Acq*Knowledge* 3.9.2 and later—Spike Location Method | Choose how spikes are searched for in the signal.<br>**Mean + Stddev**—uses fixed level peak detection with a level that is computed from the mean value plus a configurable number of standard deviations of the data.<br>**Amplitude/Half-width Discriminator**—allows for basic isolation of spike shapes that have peak voltages within a configurable range and spike half-widths within a configurable range; uses the amplitude of the spike as well as the width of the spike to determine what constitutes a valid spike event.<br>*Half-width* For a given spike, the discriminator searches from the maximum value of the spike to both the left and right of the maximum for the sample positions where the value has dropped below 50% of the maximum. The time interval between these sample positions is defined as the estimate of the half width. The acquisition sampling rate can be increased to improve accuracy of the spike half width estimates as neuron firing events involve high frequency components.<br>Each time the discriminator is run, the user must input the amplitude low and high values as well as the minimum and maximum spike width. The discriminator searches for spikes in a signal *x* as follows:<br>1. Performs regular peak detection on *x* using a fixed threshold $a_{low}$. This locates the local maxima occurring after each threshold crossing of the low amplitude area. This results in a sequence *p* of potential spike locations.<br>2. Computes the half-width time interval for each potential spike location *p*.<br>3. Accepts the spike for each potential spike location *p* as a valid spike *s* if, where *a* is amplitude and *t* is time window: $a_{low} \le x(p) \le a_{high} \wedge t_{low} \le t_{half}(p) \le t_{high}$<br>4. For each valid spike location *s*, positions the spike episode start output at *s*+*offset* and the spike episode end output at *s*+*offset*+*width* .<br>The spike discriminator generates output only for spikes that fall within the bounds of the amplitude and offset windows. If a spike candidate falls outside the windows, no output is generated. | Mean + Stddev |
| Spike Detection Level | Spikes are located in the signal by looking for locations where the signal deviates from its baseline by a certain number of standard deviations. This multiplier is set in this preference. | 4 standard deviations |
| Default Episode Width | The first time that any of the spike detection is run on a graph, the width of each fixed width episode must be specified. This preference provides the default value that is seeded in the dialog.<br>The episode width for an individual graph does not need to match this default. | 10 milliseconds |

| *Preference* | *Description* | *Default Setting* |
|---|---|---|
| Default Episode Offset | Each fixed width episode is located around one of the spikes in the signal. The offset allows for the episode to begin before (or after) the spike threshold crossing so the leading edge of the spike can be captured.<br>Negative numbers indicate episodes are to start before the spike threshold crossing, positive numbers indicate episodes that start after. | -2 milliseconds |
| Default Number of Spike Classes | The Classify Spikes script requires the user to input the number of classes into which the spikes will be partitioned. This preference allows the default number to be modified.<br>The number of classes that wind up being used does not need to match this default. | 2 |

## Principal Component Denoising




Removes noise from certain types of signals. For principal component denoising to be effective, more than two signals should be used as sources and all source channels must have identical waveform sampling rates. PCA denoising is most effective on signals that are known to contain a high degree of similarity, such as multiple ECG leads or multiple EEG leads. PCA denoising should not be used on signals of different types or units as all of the principal components may be needed to fully capture the differences in the signals.

➲ To determine if PCA denoising will be effective on a particular set of data or to compute a appropriate variance percentage for denoising, examine the principal components directly with "Transform > Principal Component Analysis" before selecting this option.

Given a set of signals, a principal component analysis is performed. The strengths of the components are then analyzed, and the original signals are reconstructed from a subset of the principal components that capture a certain percentage of the total variance of the signals. This essentially eliminates one or more of the higher-order principal components. For certain types of signals, these principal components are the ones that model the noise inherent in the signals.

1. Check the "Denoise" column for the channels to be denoised.
2. Enter the overall percentage of the variance.
3. After the percentage is entered, the denoising process will begin.

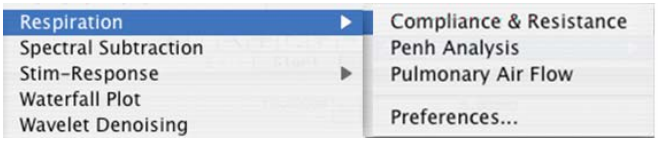
## Remove Trend

Helps to remove baseline drift or other linear trends from data. This tool makes it easier to apply trend removal to only specific segments of a waveform. Given a selected segment of data, or an entire waveform, it computes the trend between the two endpoints (similar to the Slope measurement), and then removes this trend from the selected area such that the endpoints of the selection lie at the same voltage.

Select which endpoint will remain fixed:

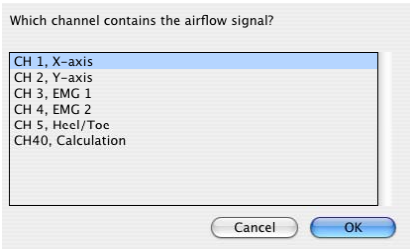
- Left keeps the starting point of the selection fixed at the same voltage. The software adjusts the data from left to right such that the right endpoint is aligned with the initial starting voltage.
- Right keeps the ending point of the selection fixed at the same voltage. The software adjusts the data from right to left such that the left endpoint is aligned with the initial ending voltage.

## Respiration

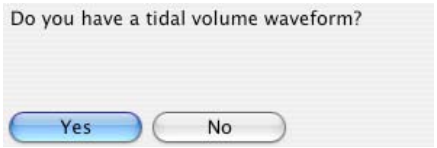


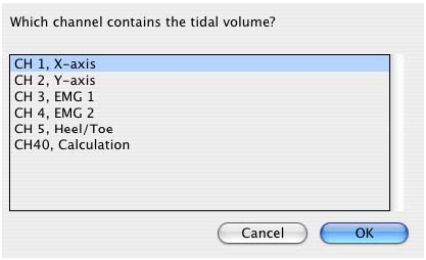

The respiration analysis package helps to analyze respiration- and airflow-related data. Other tools exist for respiration related analysis including Acq*Knowledge* transformations and the Respiratory Sinus Arrhythmia analysis in the Hemodynamics analysis package.

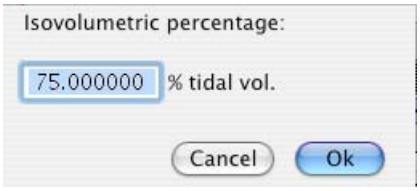
### *Compliance & Resistance*

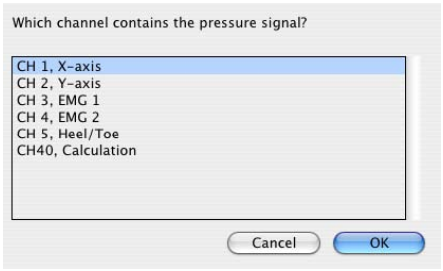







The Compliance & Resistance analysis follows the flow conventions of the recommended BIOPAC connections for a TSD107 pneumotach or a TSD117 airflow transducer. Positive flow is assumed to indicate inhalation; negative flow is assumed to indicate exhalation.

The Compliance & Resistance analysis can be used to extract pulmonary resistance and pulmonary compliance in addition to basic airflow measures. This analysis requires an airflow signal and a pressure signal. The analysis will extract all of the measures of the Pulmonary Airflow analysis for the airflow signal. It also will locate apnea periods after exhalation using the same user-configurable threshold method as the Pulmonary Airflow analysis.

Pulmonary resistance is computed using the isovolumetric method. On both sides of the tidal volume peak for a breath, the position where the volume reaches a user-specified percentage of the tidal volume is located. The pulmonary resistance is defined as the difference in pressure divided by the difference in flow at these two isovolumetric positions. Due to the discrete nature of sampled data, these points may not be exactly equal in volume. To improve the accuracy of the isovolumetric method, increase the sampling rate used to acquire data.

Dynamic pulmonary compliance is extracted on a breath-by-breath basis by dividing the tidal volume by the change in pressure between the exhale start and inhale start locations of the breath.

Individual breaths are defined as the period between consecutive Inhale Start events. Airflow units are assumed to be the standard liters/sec and pressure units mmHg. For each breath period, the analysis will define the following events:

- Inhale Start event on flow signal at start of inhale
- Exhale Start event on flow signal at start of exhale
- Apnea Start event on flow signal at beginning of apnea period (if present)
- Recovery events on volume signal at isovolumetric positions to left and right of tidal volume peak

If Inhale Start and Exhale Start events are already present on the flow signal at the start of analysis, those events will be used to define the breath periods. Apnea Start and Recovery events will always be regenerated by the analysis.

The analysis will extract the following measures from the data:

| ***Name*** | ***Abbrev.*** | ***Description*** | ***Units*** |
|---|---|---|---|
| Cycle | | Index of the breath in the data, beginning at 1. | |
| Time | | Starting time of the inhale of the breath. | seconds |
| Peak Inspiratory Flow | PIF | Maximum absolute flow occurring during the inhale portion of the breath. | liters/sec |
| Peak Expiratory Flow | PEF | Maximum absolute flow occurring during the exhale portion of the breath. | liters/sec |
| Tidal volume | TV | Total volume of air inhaled during the breath. | liters |
| Minute volume | MV | Volume of air that would be inspired during a minute given the tidal volume and breathing rate of this breath.<br>TV * BPM | liters/ minute |
| Breaths per minute | BPM | Breathing rate.<br>$\frac{60}{TT}$ | BPM |
| Inspiration time | IT | Time interval between the start of inhale and the start of exhale. | seconds |
| Exhalation time | ET | Time interval between the start of exhale and either:<br>• start of apnea (if apnea present)<br>• start of subsequent breath (if no apnea present) | seconds |
| Total breath time | TT | Time interval between the start of inhale and the start of inhale of the following breath. This is the sum of the inhalation time, exhalation time, and apnea time. | seconds |
| Apnea time | AT | Time after end of exhalation where the airflow signal remained within the apnea threshold defined at the start of the analysis. | seconds |

| *Name* | *Abbrev.* | *Description* | *Units* |
|---|---|---|---|
| Pulmonary resistance | RES | Change in pressure divided by change in flow at the isometric volume locations: $\frac{\Delta p}{\Delta f}$ | mmHg/ (liters/ sec) |
| Pulmonary compliance | Cdyn | Tidal volume divided by the change in pressure between exhale and inhale locations in the breath: $\frac{TV}{\Delta p}$ | liters/ mmHg |

If text output is being generated, an additional row will be added containing the average values of the measures. Time and count are not output as waveforms in the graph since they can be found in the horizontal axis.

### *Penh Analysis*

Penh Analysis script assumes standard recording methodology for a full body plethysmograph. Positive flow is treated as exhalation and negative flow is treated as inhalation.

Penh Analysis extracts measures from data recorded in a full body plethysmograph. It operates on a single channel of data recorded from an airflow transducer connected to the plethysmograph. The analysis takes a single parameter: the Rt percentage. This percentage is used to locate the plateau, or "pause," in the airflow signal. The pause begins at the time when the Rt percentage of the exhalation volume has been reached. The Rt percentage may be adjusted by the user and is set to a default of 65%. This analysis will place Inhale Start and Exhale Start events on the airflow signal. If these events are already present when the analysis starts, the user-defined inhale and exhale events will be used. This allows for the analysis to be repeated after manual inspection and correction of inhale and exhale locations and allows for different methods to be used to define the breathing boundaries.

Penh analysis will place Recovery events on the airflow channel at the time positions where the corresponding percentage of the volume has been exhaled. The percentage used for the analysis is displayed in the label of the Recovery events.

For each exhalation period, the Penh Analysis will extract the following:

| *Name* | *Abbrev.* | *Description* | *Units* |
|---|---|---|---|
| Cycle | | Index of the exhalation cycle in the data, beginning at 1. | |
| Time | | Starting time of the exhale for the cycle.<br>This is the location of the Exhale Start event. | Seconds |
| Peak inspiratory flow | PIF | Maximum absolute airflow occurring in the inspiration cycle immediately preceding the exhalation cycle.<br>This measure is recorded as an interval, so its value is always positive. | Airflow channel units |
| Peak expiratory flow | PEF | Maximum airflow during the exhalation cycle being examined. | Airflow channel units |

| *Name* | *Abbrev.* | *Description* | *Units* |
|---|---|---|---|
| Exhalation time | Te | Total time elapsed between the start of the exhalation cycle and the end.<br>This is the time interval between the Exhale Start and following Inhale Start events. | Seconds |
| Relaxation time interval | Rt | Time required for the subject to exhale the specified percentage of the total exhaled air.<br>This is the time interval between the Exhale Start and the subsequent Recovery event. | Seconds |
| Pause | | Numerical factor describing the characteristics of the plateau at the end of the expiration cycle. Computed using the formula:<br>$\frac{Te}{Rt} - 1$ | |
| Enhanced pause | Penh | Pause scaled to be relative to the strength of the inhale and exhale. This helps take breathing variability into account. Computed using the following formula:<br>$\frac{PEF}{PIF} * Pause$ | |

The Penh analysis excludes the following exhale cycles from the analysis:

- Exhale cycles that do not have a preceding inhale (may occur for partial cycles at the start of the data recording).
- Exhale cycles that do not have a corresponding recovery time (often occurs during apnea).

In addition, during periods of apnea, the analysis may produce invalid results, such as zero width recovery times. These results may be excluded from the analysis by either using waveform editing to remove apnea periods, discarding all events during apnea periods and rerunning the analysis, or deleting the corresponding rows from the Excel output.

### *Pulmonary Airflow*

The Pulmonary Airflow analysis follows the flow conventions of the recommended BIOPAC connections for a TSD107 pneumotach or a TSD117 airflow transducer. Positive flow is assumed to indicate inhalation; negative flow is assumed to indicate exhalation.

The Pulmonary Airflow analysis extracts basic parameters from a calibrated airflow signal, such as would be recorded using a pneumotach or airflow transducer. In addition to inspiration and expiration, Pulmonary Airflow also can be used to examine apnea. Apnea is defined in this analysis as pauses in breathing that occur after an exhalation.

When performing the analysis, an airflow signal *f* is chosen. An apnea threshold $a_f$ is also entered.

Inhalation is defined to begin at the point where $f > a_f$. Exhalation is defined to begin at the point where $f < -a_f$. Apnea is defined to be the period between exhalation and inhalation where the flow lies within the apnea threshold: $f \in (-a_f, a_f)$. At least two consecutive samples must occur within the apnea threshold for a

period of apnea to be defined. This allows for valid transitions from exhalation to inhalation to occur even if one of the samples in the transition happens to fall within the apnea threshold due to sampling.

The Pulmonary Airflow analysis will generate a tidal volume waveform if it is not present in the graph. It also will add Inspire Start and Expire Start events on the airflow signal if they are not present. New Apnea Start events will be defined each time the analysis is performed.

Individual breaths are defined as the period between consecutive Inhale Start events. Airflow units are assumed to be the standard liters/sec.

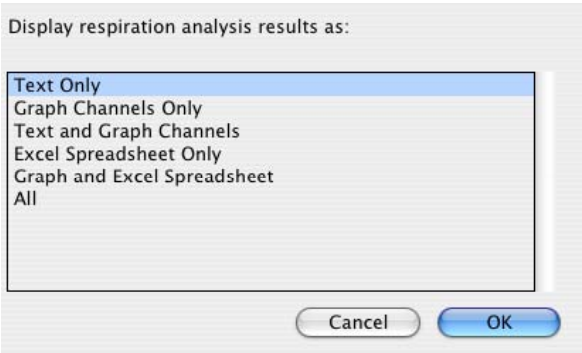
For each breath period, the analysis will extract the following:

| ***Name*** | ***Abbrev.*** | ***Description*** | ***Units*** |
|---|---|---|---|
| Cycle | | Index of the breath in the data, beginning at 1. | |
| Time | | Starting time of the inhale of the breath. | seconds |
| Peak Inspiratory Flow | PIF | Maximum absolute flow occurring during the inhale portion of the breath. | liters / sec |
| Peak Expiratory Flow | PEF | Maximum absolute flow occurring during the exhale portion of the breath. | liters / sec |
| Tidal volume | TV | Total volume of air inhaled during the breath. | liters |
| Minute volume | MV | Volume of air that would be inspired during a minute given the tidal volume and breathing rate of this breath. $TV \square BPM$ | liters / minute |
| Breaths per minute | BPM | Breathing rate for the breath. $\frac{60}{TT}$ | BPM |
| Inspiration time | IT | Time interval between the start of inhale and the start of exhale in the breath. | seconds |

| *Name* | *Abbrev.* | *Description* | *Units* |
|---|---|---|---|
| Exhalation time | ET | Time interval between the start of exhale and either:<br>• start of apnea (if apnea present)<br>• start of subsequent breath (if no apnea present) | seconds |
| Total breath time | TT | Time interval between the start of inhale and the start of inhale of the following breath. This is the sum of the inhalation time, exhalation time, and apnea time. | seconds |
| Apnea time | AT | Time after end of exhalation where the airflow signal remained within the apnea threshold defined at the start of the analysis. | seconds |

If text output is being generated, an additional row will be added containing the average values of the measures. Time and count are not output as waveforms in the graph as they can be found from the horizontal axis.

### *Preferences*




## Spectral Subtraction

Spectral subtraction is a denoising technique that operates on data projected into the frequency domain. It is frequently used in speech analysis denoising applications. Spectral subtraction examines a reference noise signal and performs a Fourier transform to get the noise frequency distribution. To denoise a signal, the Foruier transform of the signal is performed. The noise estimate frequency distribution is then subtracted from the source signal. The resulting processed spectrum with the noise frequencies removed is then reconstructed into a time domain signal using the inverse Fourier transform.

Spectral subtraction performs noise removal on the entire channel in a single Fourier transformation, which allows for denoising where the noise is stationary; there is no provision for sliding window spectral subtraction at this time.

The spectral subtraction is performed using a formula with two adjustable parameters. Given a frequency spectrum $F_{noise}$ and a mixed signal $F_{mix}$, the denoised frequency spectrum is computed using the following formula:

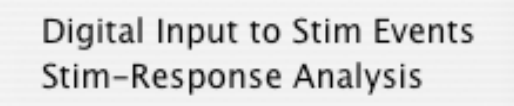
$$F_{denoise} = \left[ F_{mix}{}^{\gamma} - \alpha F_{noise}{}^{\gamma} \right]^{\frac{1}{\gamma}}$$

*where*

Alpha is the "scaling factor" and can be used to adjust the strength of the noise estimate.
Gamma is the "power factor" and can be used to vary how the noise is removed. Gamma = 1 allows for pure subtraction, Gamma = 2 allows for Euclidean distance formulas, and so on.
Any $F_{denoise}$ that is less than zero is discarded and replaced by zero to maintain a valid set of Fourier coefficients for the reconstruction.

When spectral subtraction is being used in practice, the noise signal may not always match the length of the signals to be denoised. To define the subtraction formula, the spectrum of the noise must have the same number of points as the spectrum to be denoised. If there is a length mismatch, the noise spectrum is resampled automatically to match the length of the spectrum to be denoised. Cubic spline interpolation is used during the resampling to provide a better estimate of the overall noise spectrum.

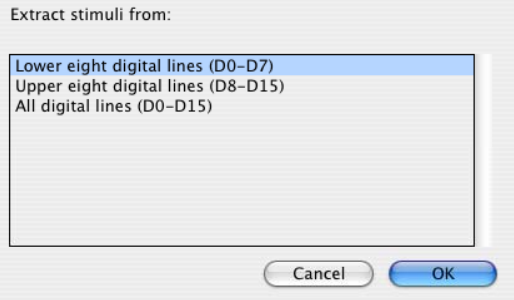
## Stim-Response




The Stim-Response analysis package can aid in analysis of stimulus-response studies. It allows for measurements to be extracted in tabular format for multiple stimulus classes.

*Note*: Excel spreadsheet generation is only supported with AcqKnowledge 3.9.1 and higher.

### *Digital Input to Stim Events*




Several stimuli delivery applications can either trigger stimulus through TTL level signals or emit synchronized TTL pulses with stimulus delivery. Examples of such software applications include SuperLab and the Inquisit programming environment.

This analysis option converts TTL data acquired on the digital channels of an MP device into stimulus events. Stimulus delivery events are defined in the graph for each low to high transition of the digital data, the indications of stimulus delivery. The digital channels are interpreted as a binary number. Each stimulus event placed into the graph has the corresponding number included with its label. This allows further analysis to distinguish between different type of stimulus events by using the Cycle Detector's label matching capability.

Digital line decoding can be two byte (using all 16 digital lines) or single byte (on either the low eight or high eight digital lines). Big endian bit and byte ordering are used, with digital line 0 representing the least significant bit.

When the stimulus labels are constructed, all numbers are zero-prefixed. All stimulus events will have the same number of base-10 digits with leading zeros, regardless of magnitude. This provides each stimulus event type with a unique label that can be used with the Cycle Detector (which uses substring matching).

Some systems that trigger digital lines such as parallel ports may not be able to do so instantaneously; they may require a time window before the transition from one state to another is fully complete. A "transition latency" time window can be given to the analysis. If non zero, any transitions that are separated by less than this latency

are treated as a single transition and only one stimulus event is inserted. The decoded value used for the transition is the maximum value observed during the transition latency window.

### *Stim-Response Analysis*

Stim-Response Analysis allows for extraction of measurements within fixed width intervals occurring at Stimulus Delivery events. The Stimulus Delivery events may be defined either manually, with the Cycle Detector, or using the Digital Input to Stim Events analysis option. The information that can be extracted includes the majority of the measurements available from the graph window measurement toolbar, matching the Epoch Analysis analysis options.

Unlike Epoch Analysis, the Stim-Response Analysis splits the analysis based upon the event labels. Stimulus Delivery events with different labels are interpreted as different stimulus types. Analysis results for each individual stimulus type are summarized in separate tables. Each independent text table has its own average of the measurements over that stimulus type.

Additional options are available for positioning the fixed width interval where measurements should be made:

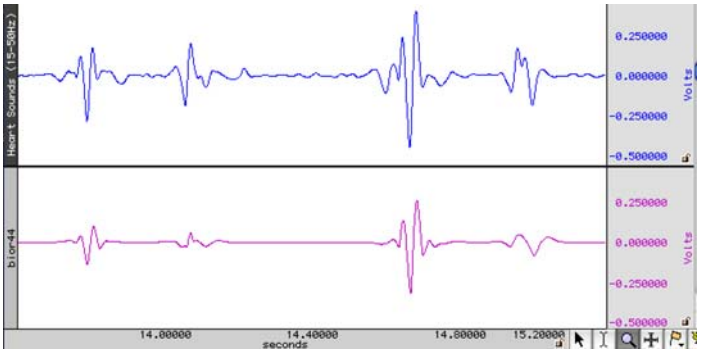
- At each stimulus event – The measurement interval is aligned so the start of the measurement matches the time of the Stimulus Delivery event.
- At fixed interval offset before or after stimulus – The measurement interval begins a fixed amount of time either before (for pre-stim studies) or after the time of the Stimulus Delivery event. This allows measurements to be made at a time relative to each stimulus onset and may be useful for measurements focusing on a specific time range (e.g. P300).
- At matching response event – This option assumes that a second set of Response events have been defined for each stimulus either manually or using the Cycle Detector. Each Stimulus Delivery event is paired with the closest Response event occurring after it. The fixed width measurement interval is aligned so the start of the measurement window is the time of this matching Response event.
  - To use the "at matching response event" window positioning option, Response events must be defined in the graph. Response events are in the "Stimulus/Response" event submenu. These events are *not* defined automatically.
    - Response events can be inserted manually into the graph using the Event tool.
    - Response events can be inserted using the event output of a data-driven Cycle Detector analysis.

*Note* If EDA/SCR signals are being analyzed in response to stimulus delivery, also examine the "Event-related EDA Analysis" transformation loctated under Specialized Analysis > Electrodermal Activity.

## Waterfall Plot

Assists in configuring the Peak Detector for 3D surface generation. These surfaces showing cycle-by-cycle data of the graph are commonly known as *waterfall plots*. Cycles are located in the graph using the same sequence of steps as the Ensemble Average transformation script. Instead of generating the averaged graph, however, a number of 3D surfaces are generated. One surface is generated for each channel that is selected by the user.

## Wavelet Denoising




*Sample output*

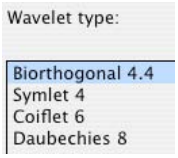
Wavelet Denoising applied to heart sounds data may help clarify $S_1$ and $S_2$, as shown:

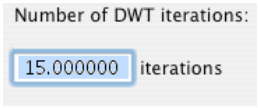
Wavelet Denoising uses the forward and reverse wavelet transformations to project source data into the wavelet domain, modify the wavelet coefficients (called "shrinking" the coefficients), and then reconstruct the data from the modified coefficients. Wavelet Denoising allows for noise to be removed from a signal while minimizing effects on portions of the signal that strongly adhere to a wavelet's shape.

To perform wavelet denoising:

1. Choose the wavelet type to use for the denoising.

   Certain signals may work best with different wavelet types.



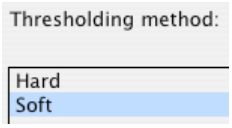



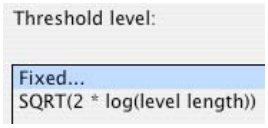

2. Enter the number of iterations to use in the wavelet decomposition.

   Different numbers of iterations will have different effects on the results.

3. Choose which type of thresholding should be used to shrink the wavelet coefficients.
   - Hard thresholding replaces coefficients below the threshold with zero while leaving all other coefficients unmodified.
   - Soft thresholding zeroes out coefficients below the threshold and subtracts the threshold for coefficients that are above it.

   Soft thresholding may be useful for reducing edge effects, but hard thresholding will affect amplitudes less.




4. Choose the threshold level to use for shrinkage.
   - Fixed threshold for all levels. If you choose a fixed threshold, an additional window will appear into which you can type your threshold.
   - Adaptive threshold level based on the number of coefficients in the DWT iteration (a VIS shrink procedure).

## ECG Analysis Algorithm References

Acq*Knowledge* 3.9 software implements the open source ecgpuwave ECG boundary location software and the open source OSEA QRS detector and beat classification library for ECG analysis.

### Automated ECG Waveform Boundary Location

➲ ecgpuwave ECG boundary location software

Acq*Knowledge* 3.9 software incorporates the ecgpuwave ECG boundary location software. Ecgpuwave is an implementation of a waveform boundary detection algorithm primarily developed by Pablo Laguna at the University of Zaragoza in Spain. This algorithm incorporates a variant of the Tompkins QRS detector, but contains additional rules that allow it to automatically extract the following characteristics of an ECG signal on a cycle by cycle basis: onset of P, P peak, end of P, onset of QRS, peak of QRS, end of QRS, onset of T, peak of T, and end of T.

The algorithm is tuned to human ECGs through comparison with manual classification. Particularly, it seems to be within the standard deviation of human examiners for the onset and end of T waves, a particularly difficult feature to extract from an ECG complex. It also has the ability to take multiple ECG leads into account to reduce errors and misclassifications and appears to function for one to twelve lead ECGs. The algorithm is well documented in a number of papers. This algorithm development was sponsored by several government agencies including CICYT in Spain and the NIH.

The ecgpuwave tool is distributed from the PhysioNet NIH servers (http://www.physionet.org). This is a tool written in Fortran that will read WFDB formatted files. It will then output a series of annotations in WFDB format indicating the locations of the various ECG complexes within each cycle. It also depends on another tool, sortann (available from PhysioNet), to perform post-processing. This software reads and writes PhysioBank formatted files. This tool is only available on OS X.
Acq*Knowledge* can automate the process of running ecgpuwave on source data and import its output back into Acq*Knowledge*. To run ecgpuwave on an ECG signal from within Acq*Knowledge*, first make the

ECG channel the active channel and then choose Transform > Specialized Analysis > Locate ECG Complex Boundaries. Acq*Knowledge* will execute ecgpuwave on that signal and read in its waveform boundary location output, placing events on the channel. You will only be able to see this output if you have events shown.

Alternatively, you can save your file to PhysioBank format, run ecgpuwave manually from a Terminal, save the annotations to “atruth” and then reopen that PhysioBank file to see the ecgpuwave results; this is the same process that Acq*Knowledge* performs.

Source code for ecgpuwave detector is released under a GPL license and can be found on the Acq*Knowledge* CD.

## OSEA QRS Detector

➲ OSEA QRS detector and beat classification library

Acq*Knowledge* 3.9 software incorporates the open source OSEA QRS detector and beat classification library.

The OSEA library is a set of routines provided by EP Limited (http://www.eplimited.com). This C++ based software library provides robust QRS complex detection and rudimentary beat classification. This library is well documented and tested. The QRS detector uses a standard Tompkins-based filtering and derivative detection algorithm and has been in development for about 15 years; the beat classifier has been developed for about a year or two. This algorithm development is sponsored by the NIH.

This algorithm is fairly robust against arrhythmias, baseline drifts, discontinuities, and other artifacts in the ECG signal. It achieves a 90% success rate on identifying QRS complexes on sample arrhythmia databases. The algorithm is tuned to human ECGs.

The QRS detector is optimized for 200 Hz sampled data. If the sampling rate is lower or higher, data will be internally resampled to 200 Hz before processing. The sampling rate difference may result in slightly different placement of beat events for different sampling rates.

QRS detection can be performed by selecting the desired channel of ECG data and choosing Transform > Specialized Analysis > Detect and classify beats. Acq*Knowledge* will execute the OSEA beat detector on the source data and output a sequence of events on that channel of ECG data. You will only be able to see this output if you have events shown.

Source code for the QRS detector is released under an LGPL license and can be found on the Acq*Knowledge* CD.

## Open Source Licensing

The ecgpuwave and OSEA algorithms are available as open source, which means that their source code is publicly available. The source code can only be used, however, under conditions of their licenses.

- ecgpuwave is under the GPL license
- OSEA is under the LGPL license

For the full text of both licenses, visit the Free Software Foundation (http://www.fsf.org).

## A Note About Specialized Analysis for Windows/PC Users

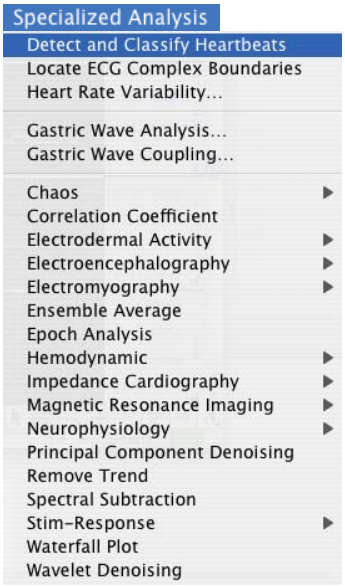
You can unlock powerful analysis tools & automation with

# Specialized Analysis
## for Acq*Knowledge* data files

Analyze data collected on MP Systems
running on Windows/PC or Mac OS X

Analyze Acq*Knowledge* data files collected on MP Systems running on Windows/PC or Mac OS X. These powerful, comprehensive analysis tools automate analysis to save hours (or days!) of processing time and standardize interpretation of results. If you need still more analysis options, save the data as MatLab, Igor Pro, PhysioNet, raw, or text format—or compress the file to reduce file size by about 60%.

For a fraction of the cost, you'll have extensive post-acquisition analysis options similar to modules from Mindware Technologies, PONEMAH Physiology Platform, EMKA Technologies, SA and other advanced analysis applications. The Specialized Analysis package is detailed on page 333.




Mac


PC/Windows

The Specialized Analysis features work with files generated by Acq*Knowledge* on a PC running Windows or on a Mac. While the data is portable, Acq*Knowledge* for Windows includes only limited Specialized Analysis tools, so you must use the Mac software to benefit from the full range of Specialized Analysis tools. Mac computers are network compatible and Ethernet ready, so it's easy to add one to an existing setup and take advantage of these powerful tools. If you're ready to add Specialized Analysis to your Acq*Knowledge* analysis options, please contact BIOPAC to discuss the possibilities and benefits of this Mac-based solution—or let us know you're interested and we'll contact you.

- To learn more about Macintosh and Mac OS 10.3 or higher, visit the Apple Store for Education or Apple® web store.

Return to Specialized Analysis Overview

# Appendix H – Network Data Transfer

Network Data Transfer allows users to access the data being acquired into a graph by Acq*Knowledge* in an external application.

The hardware API was developed to allow customers writing custom applications to control an MP unit directly and obtain data during acquisitions. Using the hardware API requires the application developer to perform appropriate configuration of the MP unit and its channels, visual graphing and data plotting, data analysis (e.g. filtering, beat detection), and persistent data storage within their application.

Some developers do not wish to undertake these additional responsibilities in their own application and prefer the ease of configuration provided by the Acq*Knowledge* graphical interface. Some development environments may also not provide enough facilities for performing all of these tasks.

For these types of situations, integration with the existing Acq*Knowledge* environment may be preferred over the more low-level hardware API.

## Objectives

- To develop a basic method for allowing third party applications to tap into the data stream being generated by both the MP unit and Acq*Knowledge* during data acquisitions.
- To provide networking facilities that allow for integration into a distributed application environment.
- To provide basic control facilities to allow external applications to query and control the Acq*Knowledge* application state.

## Overview

The real-time data transfer system is split into two types of connections:

Data connections deliver data from Acq*Knowledge* to external applications during acquisitions.

Control connections are made from external applications to Acq*Knowledge* to query application state and adjust data connections.

The *server* refers to the Acq*Knowledge* process and the computer on which it is running. The *client* refers to the custom application that is to receive data from Acq*Knowledge* and the computer on which it is running. Note that the computer used to run the Acq*Knowledge* process and the custom application may be the same computer.

All connections will be made using standard network protocols, either TCP or UDP. Single system image architectures should make connections using the loopback interface. It will be assumed that network implementations will have appropriate IP networks in place with routing between machines that can be identified either by IP address or by hostname. Firewalls must be properly configured to allow network communications between the client and server. Appropriate network configuration is beyond the scope of this document and BIOPAC Systems, Inc.

## Data Connections

Data connections are used to deliver data from the Acq*Knowledge* server to the client application. Data connections stream both data acquired from an MP unit and computed data from Acq*Knowledge* calculation channels. Data connections are available in a variety of formats and can be configured to meet the needs of the client application and bandwidth requirements.

Data connections are created only at the start of acquisitions. It is not possible to “attach” to data acquisitions that are already in progress.

XML-RPC calls for retrieving data are also described in this section. These connections are not persistent and not established from server to client, but rather a request made from client to server. The recommended mode of operation is regular TCP or UDP operation.

### Variable Sampling Rates

Acq*Knowledge* uses variable sampling rates that allow different channels of data to be acquired at different sample rates. This allows fast moving signals (such as EMG) to be acquired at high sampling rates while simultaneously recording slow moving signals (such as respiration) at lower sampling rates.

The primary use of variable sampling rate is to optimize storage requirements by minimizing space spent retaining slow moving data. In addition to the storage optimizations, variable sampling rates are also used to optimize data connections. Channels that are sampled at lower rates require less network bandwidth to transfer from client to server. If channels of data cannot be disabled and overall bandwidth is limited, variable sampling rate can be used to lower the bandwidth required to move data between the server and the client.

The use of variable sampling rates requires the client to be aware of downsampled channels and handle data accordingly. Variable sampling rate may also affect the interpretation of the incoming data connection stream, as indicated below.

### Transfer Types

Clients may choose between two different transfer types:

single connection—uses one data connection between the server and the client to deliver all data

multiple connection—uses multiple data connections delivering data simultaneously

#### *Single Connection*

The single connection transfer type uses a solitary connection between the server and the client to deliver data. When multiple channels of data are being delivered, all data will be sent over the single connection in an interleaved format. The interleaved format mingles the data of all channels in order.

- For example, an interleaved representation of two samples from three channels of data would be:

  C1 C2 C3 C1 C2 C3.

  This is equivalent to two sample frames where a *frame* contains the data at a particular sample location for all of the channels, in this case “C1 C2 C3”.

Frames are delivered sequentially over the data connection.

The single connection transfer type allows for all data to be processed from a single location and uses only one network connection resource. Single connection transfer has drawbacks including the need for the client to demux the data stream as well as account for variable sampling rates.

#### Variable Sample Rate Considerations

When variable sampling rate is being used, the single connection transfer type has the characteristic where not all frames have the same size. If a frame occurs at a sample position where a channel is not being sampled, that channel will not be included in the frame.

All downsampling within the Acq*Knowledge* variable sampling rate environment occurs at integer divisions of the base sampling rate. Only frames with indexes (zero-based) that are evenly divisible by the downsampling divider will contain data for that channel.

> **It is the responsibility of the client to compute and retain the frame index and to properly check channel dividers against the frame index to determine which channels are represented in that frame.**

Frame 0 always contains a data point from every channel.

- For example, consider a three channel acquisition. Channels 1 and 3 have a downsampling divider of 1, that is, they are being acquired at full speed. Channel 2 has a downsampling divider of 2, that is, channel 2 is being acquired at half speed. The first five frames of data in the single connection transfer type will appear as follows:

C1 C2 C3 | C1 C3 | C1 C2 C3 | C1 C3 | C1 C2 C3

Note how frames 1 and 3 are shorter as they occur at positions where channel two is not defined.

If variable sampling rate is being used, clients are required to perform proper frame indexing in order to demux the interleaved data stream. If clients cannot handle variable sampling rate correctly and are using the single stream transfer type, clients should check all of the downsampling dividers using the control connection to Acq*Knowledge* and warn the user if the configuration cannot be supported with that client.

### *Multiple Connection*

The multiple connection transfer type uses a single connection from server to client for each channel of data that is being delivered. Using multiple connections offers the benefit of avoiding the client having the need to demux all of the data from each sample. If the client is operating in a high-load environment, the elimination of the demuxing may be useful in reducing processor overhead. It also allows client code to be simpler if variable sampling rate is being used.

The primary disadvantages of using the multiple connection transfer type are the usage of more network ports and the client assuming responsibility for synchronization of data across multiple channels. The data immediately available on one network connection may not be guaranteed to be the identical sample index of data being received for another channel. The client must keep track of sample index on an individual channel basis to properly synchronize data across multiple channels.

#### Variable Sampling Rate Considerations

Using variable sampling rate in a multiple connection transfer type is significantly easier. Each individual channel's connection delivers data at that channel's sampling rate. On a given connection, all samples have an identical length.

Clients still must be aware of variable sampling rates. The sampling rate of information on a downsampled channel connection will be different than the sampling rate of information on an upsampled channel connection. If the client is performing any time domain measurements or other computations involving the sample interval, the difference in inter-sample-interval must still be taken into account.

### *XML-RPC*

The XML-RPC transfer type allows clients to explicitly request data from the server. Instead of data being automatically pushed to clients, the client must post an XML-RPC function call to the server. This will allow the client to query the server for the most recently acquired data sample value for a particular channel. This communication method is for clients that do not require continual data streams or that interact with only slow moving data. This method returns values only; no information about sample indexes or lengths is returned.

XML-RPC has significant overhead for both client and server, so this transport method cannot handle more than a few requests per second. If faster response time is required, the client should implement either the single connection or multiple connection streaming methods.

XML-RPC is not a true data connection as it does not involve the server constructing a streaming connection to the client.

## Transport Protocol

Data connections will offer a choice of using either TCP or UDP as the transport protocol for delivering data to the client. Choice of protocol depends on application requirements. When a client is receiving data, it is assumed that all data connections are using the same transport protocol.

### *TCP/IP*

TCP is the preferred transport protocol. As TCP guarantees reliable, ordered delivery, all data is simply transferred from the server to the client without any additional information. Data will be streamed continuously as it becomes available. TCP is recommended for all clients that require a guarantee of receiving all information. It is also recommended for any configuration using up to two computers.

The port number used for data connections is specified by the client using the control connection prior to the start of acquisition. Once a client passes along port information, the client should begin listening for connections on that port.

When using TCP data connections, the start of acquisition is signaled to clients by the establishment of a connection on an appropriate port to the client. The end of an acquisition is signaled by the termination of the connection.

### *UDP*

UDP is a connectionless protocol that does not guarantee either delivery or properly ordered reception of packets by clients. Data connections will be allowed to be switched to UDP delivery mode. The primary benefit of using UDP datagrams is that a single data stream can be multicast to a number of computers. Multicasting is not offered implicitly by the Acq*Knowledge* data connection protocol but can be achieved implicitly by requesting a data connection be bound to a broadcast address.

Since UDP is unordered, datagram delivery of data will include an additional four bytes prior to the data contents of the packet. These four bytes will indicate the starting sample position of the samples (or frames for single connection transfer type). The four bytes will always be in network order to allow for platform flexibility. If the client will perform interpretation of the sample index, `htonl()/ntohl()` should be used to convert from network order into host endian.

As sample positions are always monotonically increasing, the sample index tag can be used to reassemble datagrams into correct order, if required. The sample index will still need to be swapped to host order from network order for the sequence to be increasing in magnitude.

There are no provisions for clients to request retransmissions of packets from the server. Clients should be aware that data may not be delivered when using UDP and should be prepared to examine sample indexes at the beginning of each packet and handle missing data accordingly (padding, warning user, etc.).

The port on which datagrams will be delivered is specified by the client prior to the beginning of acquisition on the control connection. Once the client specifies its port, it should begin listening for datagrams delivered to that port. The start of acquisition will be signaled by the first datagram that is sent to that port. The sample index of the first datagram will correspond to the first hardware sample of data acquired by Acq*Knowledge*. If the graph is initially empty, the index will be zero.

Unlike TCP connections which get explicitly disconnected at the end of acquisitions, there is no direct messaging for UDP delivery indicating the end of acquisitions. Datagrams will not be sent to the client after the end of acquisition. Clients requiring explicit termination notification should either use TCP or implement a timeout mechanism combined with an XML-RPC getAcquisitionInProgress call on the control connection to locate the end of acquisitions.

### *XML-RPC*

The XML-RPC “get most recent data sample” call will use standard XML-RPC connection semantics involving the client making an HTTP POST request to the server and interpreting the response as appropriate. Handling of XML-RPC can be performed by a library in the client's appropriate implementation language.

## Real-time Delivery Guarantees

No delivery time guarantees exist for any data connection. Data is delivered to the client as it becomes available. The overall latency of the system from physical sample time to data delivery is dependent on a number of factors including network load, Acq*Knowledge* overhead (which is variable due to calculation channel processing time, user interface activity, thread scheduling, and other operations), system load, and other factors. The actual sampled data itself is guaranteed to be accurate; the sample time of a signal acquired into the MP unit is always accurate. It is only the time between the physical time corresponding to a sample and its delivery to the client application that is variable. If strict real time guarantees or more predictable latencies are required, users may need to consider using the hardware API on their local machine.

### Data Formats

It is assumed that the data transfer feature will be used in a mixed host environment, potentially with clients running in environments that have restricted data types. The sampled information delivered by a data connection will be allowed to be controlled to appear in a variety of different formats for the client:

- 64 bit floating point—always available for all channels and may be delivered in either big endian byte order or little endian byte order. This is equivalent to a C style double data type.
- 32 bit floating point—always available for all channels and may be delivered in either big endian byte order or little endian byte order. This data type is not native to Acq*Knowledge*, so the precision of received data may differ from data as recorded by Acq*Knowledge*. This is equivalent to a C style float data type.
- Signed 16 bit integer—available only for analog data channels. It may be delivered in either big endian byte order or little endian byte order. This is equivalent to a C style short data type. Clients receiving data in 16 bit integer format should use the control connection to determine appropriate floating point scaling factors that need to be applied to the data to convert into actual units.

Data formats should be specified by the client prior to the start of acquisition. If left unspecified, the default data format depends on channel type:

| *Channel Type* | *Default Data Type* |
|---|---|
| Analog 16 bit signed integer | little endian |
| Digital 16 bit signed integer | little endian |
| Calculation 64 bit floating point | little endian |
| XML-RPC get most recent sample | requests will always return the double type of XML-RPC (in ASCII notation) |

### Default Data Connection Settings

If the client does not modify any data connection settings, the following will be used:

- Single connection transfer type
- TCP/IP transport protocol
- Port 15020 for single connection
- For multiple connections, each channel type (16 channels per type) will be set as follows:
  - analog channels [15020-15035]
  - digital channels [15040-15055]
  - calculation channels [15060-15075]
- Default data formats for each channel type as indicated in the “Data Formats” section

Note that no channels are enabled for delivery by default. Both data connection delivery and get most recent sample tracking are disabled. Clients will still need to enable channels in order to receive data.

### *Locating AcqKnowledge Servers*

It is possible that clients and servers may be located on networks with dynamic IP addresses or other feature that make establishing the connection between machines difficult. In this case, a simple UDP broadcast mechanism can be used to locate known servers.

Acq*Knowledge* will listen for incoming UDP packets on port 15012. If the UDP packet contains the sequence of ASCII characters “AcqP Client” and only that sequence, it will send a broadcast packet containing “AcqP Server Port:” followed by the port number on which the server is listening for control connections. The port number will be expressed in base 10 ASCII notation.

It will be possible to configure Acq*Knowledge* to not ac*Knowledge* discovery requests for security purposes.

### *Control Connections*

Clients connect to an Acq*Knowledge* server using control connections. A control connection allows the client process to control how data is going to be delivered to it, query settings, modify settings, and perform other basic operations without requiring graphical interaction with the AcqKnowledg environment.

On the start of an acquisition, all data connections are established to the client that most recently established a control connection.

It will be possible to configure Acq*Knowledge* to not respond to any control connections for security purposes.

The majority of all control connections use XML-RPC. The XML-RPC specification can be located at http://www.xmlrpc.com/spec. XML-RPC consists of an HTTP POST request being assembled by the client along with data content expressed in the XML-RPC notation. The remote procedure call will then return with an appropriate response to the caller. By design, XML-RPC is client and platform agnostic.

XML-RPC implementations exist for a number of languages. If there is no implementation in the client's language, hardcoded requests can be embedded or a small helper application or shared library can assist in providing the control connection interface.

The URI that should be used when connecting to the server is the recommended "/RPC2" for XMLRPC.

- For example, a URL for localhost control connections would be "http://localhost/RPC2".

### TCP Port

Only one control connection may be opened by a client to a server at a time. The server will listen on port 15010 by default. As this port may conflict with other network services or may need to be modified for firewall accessibility, Acq*Knowledge* will allow the server port to be modified by the user in the Display > Preferences panel. Clients should be aware that control connection port numbers are not fixed and should either allow users to change the port number from the client or use the dynamic discovery mechanism for locating Acq*Knowledge* servers.

### Control Procedure Calls

Control procedure calls are remote procedure calls with a set method name and response. All method names and strings are case sensitive. The available control calls are roughly split up into querying acquisition parameters, configuring data delivery, and limited calls for modifying acquisition parameters and affecting application state.

### *Channel Index Parameter Structures*

Some procedure calls take a *channel index structure* as a parameter. This is an XML-RPC structure that consists of the channel type and index. The channel type is a string member named "type" that is one of the following strings: analog, digital, calc. The channel index is an integer member named "index" and contains a zero-based index indicating the channel. For example, the following channel index parameter structure can be used to refer to calc channel 2 (recall, in Acq*Knowledge* calc channels are indexed from zero):

```
<struct>
      <member>
            <name>type</name>
            <value><string>calc</string></value>
      </member>
      <member>
            <name>index</name>
            <value><int>2</int></value>
      </member>
</struct>
```

### *Querying Acquisition Parameters*

The calls for querying acquisition parameters are intended to allow clients to request information required for them to fill out appropriate parameters and to verify that previous control requests have been properly applied. The following control calls are recognized:

#### `getMPUnitType`

Method name: acq.getMPUnitType
Parameters: None
Return value: int

Retrieves the type of MP unit to which the server is connected. This may be zero (indicating no unit is connected, e.g. "No Hardware" mode) or one of the following: 100, 150, 35, 45. The client can use this value to decide between appropriate templates to download or other channel settings.

#### `getEnabledChannels`

Method name: acq.getEnabledChannels
Parameters: string
Return value: array populated with int

Retrieves the channels that are available for acquisition and data delivery over a data connection to the client. The type of channels that are to be returned are specified in the string parameter. The string parameter may be one of the following: analog, digital, calc. The type must be lowercase.

"analog" returns information about analog channels
"digital" digital channels
"calc" calculation channels

The enabled channels are returned as an array of channel indexes, zero-based. These are the indexes that can be validly used in calls for configuring data connection parameters.

#### `getChannelScaling`

Method name: acq.getChannelScaling
Parameters: channel index parameter structure
Return value: struct containing scaling

For channels that can be delivered in short integer format, the channel scaling provides the scale factor and offset used to convert the short into physical units using the formula: sample*scale+offset.

The scaling is returned as a structure containing two double-valued members: "scale" contains the multiplication factor, "offset" contains the offset factor.

Channels that cannot be delivered as short integers cannot be scaled. If this method is called with an invalid type of channel, a fault response will be returned.

#### `getSamplingRate`

Method name: acq.getSamplingRate
Parameters: None
Return value: double

Returns the current sampling rate expressed in Hz. This is the rate at which data is sampled for a channel whose downsampling divider is 1.

#### `getDownsamplingDivider`

Method name: acq.getDownsamplingDivider
Parameters: channel index parameter structure
Return value: int

Retrieves the downsampling divider for a particular channel. A channel is sampled at the base sampling rate divided by this factor. This factor is also used to determine which frames contain samples for this channel.

### *Data Connection Configuration Commands*

While it is possible to configure data connection parameters using Acq*Knowledge*, frequently clients may need to alter data connection configurations based upon dynamic information regarding the machine on which they are running (e.g. port collisions with other services, unexpected endian changes, etc.). The following commands may be used to configure how data is delivered to the application.

#### `getDataConnectionMethod`

Method name: acq.getDataConnectionMethod
Parameters: none
Return value: string

Returns the method currently being used for data connections between the server and client. The string value is either: single, multiple.

"single" corresponds to a single data connection being made between the server and the client with all data interleaved over that connection.

"multiple" corresponds to opening an individual connection to the client for each channel.

#### `changeDataConnectionMethod`

Method name: acq.changeDataConnectionMethod
Parameters: string
Return value: 0 on success, or fault code

Changes the method used to deliver data to the client. The parameter is a string that is one of the following: single, multiple.

"single" opens up a single data connection and sends data in an interleaved fashion.

"multiple" opens up an individual data connection for each channel.

#### `getTransportType`

Method name: acq.getTransportType
Parameters: None
Return value: string

Retrieves the transport type that is being used to deliver data from the server to the client. The transport type is a string that is one of the following: tcp, udp. Note that the XML-RPC data delivery method may be used in addition to this transport type if channels are enabled.

#### `changeTransportType`

Method name: acq.changeTransportType
Parameters: string
Return value: 0 if successful, else fault code

Change the transport type that is used to deliver data from the server to the client. The transport type is a string that has one of the following values: tcp, udp. XML-RPC last value data delivery may be used in addition to this type provided channels are enabled properly.

#### `getUDPBroadcastEnabled`

Method name: acq.getUDPBroadcastEnabled
Parameters: None
Return value: boolean

Determine if UDP packets are sent only to the client or are broadcast to the broadcast IP of the network. Broadcasting is supported only when the transport type is UDP.

### changeUDPBroadcastEnabled

Method name: acq.changeUDPBroadcastEnabled
Parameters: boolean
Return value: 0 if successful, fault code on error

Modify whether UDP packets are sent only to the client or are broadcast to the broadcast IP of the network. Broadcasting is supported only when the transport type is UDP.

### getSingleConnectionModePort

Method name: acq.getSingleConnectionModePort
Parameters: None
Return value: integer

Returns the port number on which the server will connect to the client to deliver data. This port is used only when the connection mode is set to “single” which interleaves all data over a single connection.

### changeSingleConnectionModePort

Method name: acq.changeSingleConnectionModePort
Parameters: integer
Return value: 0 on success, else fault code

Modifies the port on which the server connects to the client to deliver data. This port is used only when the connection mode is set to “single” which interleaves all data over a single connection.

### getDataDeliveryEnabled

Method name: acq.getDataDeliveryEnabled
Parameters: channel index parameter structure
Return value: boolean

Query whether a channel is enabled for data delivery. Channels must be enabled for data delivery in order for their data to be delivered to the client. Not all channels that are being acquired are required to be delivered to data delivery.

### changeDataDeliveryEnabled

Method name: acq.changeDataDeliveryEnabled
Parameters: channel index parameter structure, boolean
Return value: 0 for success, else fault code

Change whether or not data delivery is enabled for a particular channel. Data delivery can only be changed prior to the start of an acquisition. Changes to data delivery enabling are only applied on the next start of acquisition.

### getMostRecentSampleValueDeliveryEnabled

Method name: acq.getMostRecentSampleDeliveryValueEnabled
Parameters: channel index parameter structure
Return value: boolean

Query whether a channel is enabled for most recent data sample requests. If a client wishes to use the XML-RPC calls to fetch the most recent value of data acquired of a channel, the channel must be enabled for this functionality prior to the start of acquisition.

### changeMostRecentSampleValueDeliveryEnabled

Method name: acq.changeMostRecentSampleDeliveryValueEnabled
Parameters: channel index parameter structure, boolean
Return value: 0 for success, else falut code

Change whether or not a channel is enabled for most recent data sample requests. When a channel is enabled, XML-RPC calls can be used during an acquisition to return the most recent sample of data acquired (or computed) for the channel. Any changes to the enabled state of a channel will be applied on the start of the next acquisition. If a client wishes to use XML-RPC calls to read the value of a channel, that channel must be enabled prior to the start of the acquisition.

### getDataConnectionPort

Method name: acq.getDataConnectionPort
Parameters: channel index parameter structure
Return value: int

Retrieves the port on which the the server will deliver the data for the channel specified in the parameters to the client. Per-channel data connections are only used if the data connection method is set to “Multiple”.

### changeDataConnectionPort

Method name: acq.changeDataConnectionPort
Parameters: channel index parameter structure, integer
Return value: 0 for success, else fault code

Changes the port on which the individual connection is made by the server to the client to

deliver the data for the channel specified in the parameters. This style of connection is used only if the data connection method is set to “Multiple”.

### getDataType

Method name: acq.getDataType
Parameters: channel index parameter structure
Return value: structure

Returns the data type that is being used in the binary data streams for the channel's data. The return value is a structure with a type and endian member. The type member, named “type”, is a string and contains one of the following values: short, double, float. These strings correspond to their matching C-style data types. The endian member, named “endian”, is a string and contains one of the following values: little, big. “little” corresponds to little endian byte order, big endian bit order. “big” corresponds to big endian byte order, big endian bit order.

### changeDataType

Method name: acq.changeDataType
Parameters: channel index parameter structure, type structure
Return value: 0 on success, or fault code

Changes the data type that is used for binary data streams of the channel's data. The type structure is a struct containing two members. The “type” member is one of the following strings: double, float, short. Each string corresponds to the matching C-style data type. The “endian” member is one of the following strings: little, big. Each corresponds to the matching byte endian. Bit order within a byte will always be big-endian. Not all channels may be able to support all data types. If the channel cannot be transmitted in the requested data type, a fault code will be returned.

### *Reading Data During Acquisition*

Normally data is delivered to clients during acquisitions using data connections. Data connections are either TCP or UDP connections established from the server to the client over which the server streams the incoming data. In some languages and environments however, it may not be possible to handle continuous data streams. The following XML-RPC commands are offered to assist these types of clients. Due to the large overhead of processing connections and XML-RPC requests, this data transfer type is not recommended and TCP/UDP should be used wherever possible.

#### `getMostRecentSampleValue`

Method name: acq.getMostRecentSampleValue
Parameters: channel index structure
Return value: double, or fault code

This procedure allows clients to read the most recent data value of a specific channel during acquisitions. In order for this call to be successful, a data acquisition must be in progress and the channel must be enabled for most recent sample value data delivery. Issue a changeMostRecentSampleValueDeliveryEnabled call prior to acquisition to allow this procedure to be used.

#### `getMostRecentSampleValueArray`

Method name: acq.getMostRecentSampleValueArray
Parameters: none
Return value: array of structures with channel info and values, or fault code

This procedure allows clients to retrieve the most recent data values of all channels during acquisitions in a single call. If clients are interested in the values of multiple channels, using this method is more efficient than performing consecutive getMostRecentSampleValue calls (which require an individual POST request per call).

Only channels that are enabled for most recent sample value data delivery will be returned.

Issue a changeMostRecentSampleValueDeliveryEnabled call for each desired channel prior to acquisition in order for this to return the value for a channel.

The return value is an array of structures, one per channel. Each structure contains two members.

"channel" member contains a channel index structure with members set to appropriate type and index information for the channel.

"value" members is a double that contains the most recent sample value of the channel specified in the "channel" member.

This procedure cannot be called unless there is an acquisition in progress and there is at least one channel enabled for most recent sample value delivery.

### *Other Control Connection Commands*

Control connections will also allow for the following additional commands to be used by clients:

#### `loadTemplate`

Method name: acq.loadTemplate
Parameters: base64 encoded binary AcqKnoweldge graph template
Return value: 0 on success, or fault code

Attempts to open the passed template within the Acq*Knowledge* environment. The parameter is a base 64 encoded Acq*Knowledge* "gtl" graph template file. This command can only be used with templates in Acq*Knowledge* 3.7.1-Windows or Acq*Knowledge*-Mac graph templates. The parameter must include the entire contents of the template file.

The parameter in the XML-RPC call should be a base64 parameter with the raw binary contents of a graph template file. When this parameter is decoded, it should correctly provide the contents of a template file on disk.

This function will return 0 on success; otherwise a fault code if the template could not be loaded. Once a template is loaded, the hardware settings contained within that template will be used for subsequent data acquisitions. Templates for Acq*Knowledge* 3.9.2-Mac and higher will retain any data connection settings that have been specified by clients. After loading a template, clients should re-send any configuration information for ports and data connection methods if they do not match the new settings from the template.

If the template data is corrupted or is incompatible with Acq*Knowledge*, user interaction may be required on the server computer to dismiss any error messages when attempting to load the template. If user interaction is required on errors, this call may not return until the user interaction has completed.

Clients who are using XML-RPC bindings that offer timeout services may wish to use them with this function.

#### `getAcquisitionInProgress`

Method name: acq.getAcquisitionInProgress
Parameters: none
Return value: boolean

Query whether data acquisition is currently in progress or not. A value of true is returned if data acquisition is occurring in any open Acq*Knowledge* graph window.

#### `toggleAcquisition`

Method name: acq.toggleAcquisition
Parameters: none
Return value: 0 on success, else fault code

Toggles data acquisition in the frontmost graph. If data acquisition is in progress, it is halted. If none is in progress, data acquisition is started in the graph.

Note that this function invocation may block if physical user interaction is required to start the acquisition in the graph, such as dismissing an overwrite warning, warnings on incompatibilities between different MP unit types, specifying a save location for acquisition to disk, etc. If the implementation of the XML-RPC binding used by the client supports timeout capabilities, it is highly recommended to enable timeouts for this function.

### Implementation Notes

The network data transfer feature will be incorporated into Acq*Knowledge* as a new realtime processing object invoked in fashions similar to audio playthrough engines. The application will be extended with third party libraries to allow for XML-RPC operation. New preemptive threads will be added to the application to handle server discovery requests, control connection requests, and data transfer during acquisitions.

#### *External Libraries*

Acq*Knowledge* will use external libraries to assist in implementing the ability to perform network data transfer and respond to XML-RPC requests.

#### xmlrpc-c

The XML-RPC layer will be implemented using the xmlrpc-c library (http://xmlrpc-c.sourceforge.net).

This library is a small layer that allows for implementation of both clients and servers in C and C++ applications. For servers, the library also includes a version of the abyss web server that will handle listening for connections and can use a number of external libraries for posting new XML-RPC requests.

xmlrpc-c will be kept in the Other subdirectory of the repository. An Xcode project will be made for xmlrpc-c to allow it to be included directly as a dependency in other projects. To facilitate builds, the default configuration will be renamed to "Imported CodeWarrior Settings". The output will be a series of static Universal binary libraries that can be linked directly into applications.

On Mac OS X, the libcurl library is provided at the system level. xmlrpc-c will be configured to use libcurl to perform its client functions.

### *Licensing*

xmlrpc-c is licensed using a variety of licenses including the main xmlrpc-c license, the expat license, the python license, and the abyss web server license. All of these licenses allow for direct linking of the library with a commercial application. Appropriate copyright notices will be added to the About dialog and into the documentation, where appropriate.

### *Threading Constructs*

Wherever possible, the implementation of the network data transfer feature will use preemptive threads. The Carbon Multiprocessing library will be used to spawn the threads. This will allow the threads to use MPRemoteCall to schedule functions for execution on the main thread. Main thread scheduling will be required for any operations that have application state side-effects for non mutex guarded structures or operations that result in user interface change.

### *Network Configuration Storage*

The main preferences dialog accessed through Display > Preferences will allow the user to toggle network access on and off, change the default control connection TCP port, and disable automatic server discovery. These settings will be stored in the user preferences file and will be modifiable on a per user-account basis.

Additional network configuration storage will be stored in a new structure incorporated into the hardware acqStruct. This storage will include the data transfer settings (both global and channel specific), the data type requested for each channel, and other configuration. These settings will be retained in the creator handle and graph files. No visible user interface will be added for these settings and they can only be modified by a client using the XML-RPC requests over the control connection.

The connection settings may be different from graph to graph and can be modified after loaded from a graph file. This allows clients to send over a template with the majority of their settings already intact, leaving clients to change only specific dynamic values such as client IP address.

Since the network configuration storage settings may be accessed from multiple preemptive threads, a preemptive safe mutex will be used to guard all access to the settings from all threads. Additionally, switching of the hardware parameters as different graphs are brought to the front will also respect this mutex and not change any values while they are being read by another thread.

### *Control Connection Server*

Control connections are made from clients to the server through XML-RPC requests. Acq*Knowledge* will implement the control connection server using the xmlrpc-c library with the built-in abyss web server used to listen for connections. To improve both response time for both clients as well as the Acq*Knowledge* application, the server listening for control connections will be implemented on its own preemptive thread. The server will use the runConn() method of the C++ server class while handling its own socket generation with accept in order to keep track of the most recently connected client.

While this limits processing to one RPC at a time, it is the only method that will keep the RPC invocations within the Acq*Knowledge* address space, required to modifiy settings of in-memory graph structures. If the network server is enabled in the user preferences, the thread will be created when the application is launched and will persist until the application exits. Any changes to listening port for control connections in the application preferences will cause the thread to be stopped and recreated to listen on the new port.

The control connection protocol does not include any type of graph identifier for each call. The procedure calls that are available will always refer to the topmost graph window. There will be no way for clients to change the top graph remotely.

An exception to the graph ordering is the “loadTemplate” command. When issued, the data sent will be decoded by Acq*Knowledge* and will be used to construct a new topmost graph. As the control connection thread is preemptive, this process will be scheduled on the main thread. The XML-RPC call will be treated as blocking and will not return until either the template has opened or the main thread has delivered an error code back to the service. This cross-thread call will be handled at the same location as the scheduling of macro invocations on the main thread.

The toggleAcquisition XML-RPC command will be similarly scheduled onto the main thread as `ToggleAcquisition()` assumes the cooperative threading environment for proper data structure access restrictions.

### *Server Discovery Thread*

The design includes a rudimentary method that clients can use for locating Acq*Knowledge* servers on their LAN. It consists of listening for UDP broadcasts and replying with appropriate UDP broadcasts, similar to the MP150 discovery mechanism. If the server feature is enabled when the application is launched, a new preemptive thread will be spawned to respond to discovery requests. The preemptive thread will live for the duration of application execution while networking is enabled and will be terminated on application exit.

### *Network Data Transfer Engine*

The network data transfer engine will be implemented as a new NetworkDataTransferEngine object in corresponding files. This engine will have a `ProcessIncomingData()` method that will be invoked during acquisitions at the same time as audio playthrough. It will examine the graph structure directly, keeping track of buffer lengths of channels inbetween successive calls. When new data is encountered, it will be copied onto a second set of buffers that are private for the network data transfer engine and onto an array holding the most recent sample values.

Similar to the audio processing engines, there will be one network data transfer engine created for each hardware structure and will live for the lifetime of the structure. The engine will use the AcquisitionEventListener framework to respond to starts and stops of acquisitions.

At the start of acquisitions, the data transfer engine will be configured to match the network transfer settings as currently stored in the hardware structure. These settings will not be allowed to be modified while the acquisition is in progress.

After initialization, the engine will attempt to establish any requested data connections as BSD sockets, either TCP or UDP as configured by the client. Once all connections are successfully established, a preemptive thread will be spawned to handle the actual data transfer. This thread will examine the private network data transfer buffers. Whenever new data is encountered in these buffers, the preemptive thread will assemble that data for delivery to the client. A single preemptive thread will handle all connections. The preemptive thread will be killed at the end of the acquisition, and any system resources will be gracefully released prior to the thread exit.

To support the XML-RPC get most recent data sample functionality, the engine will maintain an array of doubles with the most recent sample that was copied into the network transfer buffers. This double valued array can be accessed by other threads to obtain the sample values. The actual XML-RPC calls for returning the most recent value will be handled by the XML-RPC thread and not the network data transfer thread.

All buffers that are allocated will be circular buffers and will not be reallocated during the lifetime of the thread. The `ProcessIncomingData()` method called from the acquisition thread will be the producer and the only modifier of the head pointer. The preemptive thread will be the consumer and the only modifier of the tail pointer. This one to one producer/consumer arrangement alleviates the need to perform any mutex locking on the buffers.

## ACKNOWLEDGEMENTS

Technical Writer: Jocelyn Mariah Kremer with input from Alan Macy, Frazer Findlay, Edward Peterlin, and Kelly Stephens.

Cover and frontispiece illustrations: Creative Resource Group, Santa Barbara, CA.

Created with Microsoft Word for Windows, JASC, Inc. JasCapture, Adobe Photoshop and Corel Draw 7.

Manual Revision: AcqKnowledge 3.9.2 for Mac (3/07); network data transfer (8/07), SCR $t_{onset}$ (2/08).

PDF 02/19/08