ECG_BASELINE

Filter_Params

Filter coeffs
Length

Filter coeffs calculation

<<Enumeration>>
Filter_Type

moving_average
butterworth
non-adaptive
adaptive

Ecg_Baseline_Module

signal_raw: double
signal_filtered: double
signal_baseline: double
sampling_frequency: double
time_vec: double
filter_type: enum
filter_params: class

load_signal(): void
filter_set_properties(): void
filter_noise(): void
filter_baseline(): void
get_signal_raw(): double

get_signal_filtered(): double
get_signal_baseline(): double
get_time_vec():double
filter_moving_average(): private void
filter_butterworth(): private void
filter_non_adaptive(): private void
filter_adaptive(): private void

R_PEAKS

<<Enumeration>>
R_Detection_Method

pan_tompkins
hilbert_transform

R_Peaks_Module

signal_filtered: double
sampling_frequency: double
time_vec: double
r_detection_method: enum
r_peaks: double

find_r_peaks(): void
get_r_peaks(): double
pan_tompkins(): private void
filter_bandpass(): private void
filter_lowpass(): private void
filter_highpass(): private void
differentiate(): private void
square(): private void
integrate(): private void
hilbert_transform(): private void

HRV2

Histogram

bins:
values:

Setting
Getting

Other_Classes

Params
SD1: double
SD2: double

Setting
Getting

Hrv2_Module

r_peaks: double
time_vec: double
cum_time_vec: double
histogram: class
tinn: class
triangular_index: class
poincare: class

calc_cum_time_vec(): void
calc_histogram(): void
calc_tinn(): void
calc_triangular_index(): void
calc_poincare(): void
calc_SD1(): private void
calc_SD2(): private void
get_hist(): class
get_tinn(): class
get_triang_index(): class
get_poincare(): class

HRV1

Hrv1_Module

r_peaks:double
time_vec: double
cum_time_vec: double
uniform_time_vec: double
frequency_vec: double
periodogram: double
time_params: class
frequency_params: class

calc_cum_time_vec(): void
resample(): void
calc_freq_vec(): void
calc_periodogram(): void
calc_time_params(): void
calc_freq_params(): void
%some helpful methods
get_periodogram(): class
get_freq_vec(): double
get_time_params(): class
get_freq_params(): class

Time_Params

time_param1: double
time_paramn: double

Setting params
Getting params

Frequency_Params

ULF: double
VLF: double
LF: double
HF: double
freq_ULF:
freq_VLF:
freq_LF:
freq_HF:

Setting params
Getting params

WAVES

Waves_Module

signal_filtered:double
sampling_frequency: double
time_vec: double
waves: class

find_waves(): void
get_waves(): class

Waves

r_peaks: double
qrs_onset: double
qrs_end: double
t_end: double
p_onset: double
p_end: double

Setting points
Getting points

HRV_DFA

Hrv_Dfa_Module

r_peaks: double
params: class
plots: class

set_r_peaks(): void
calc_params(): void
calc_plots(): void

Hrv_Dfa_Params

Params

Setting
Getting

ST_SEGMENT

St_Segment_Module

waves: class
signal_raw: double
signal_baseline: double
params: class

set_waves(): void
set_signal(): void
analyze(): void

St_Segment_Params

Params
Setting
Getting

HEART_CLASS

Heart_Class_Module

waves: class
signal_raw: double
signal_baseline: double
heart_class: enum - depends on library

classify(): void
get_heart_class(): enum

<<Enumeration>>
Heart_Classs

N
A

T_WAVE_ALT

T_Wave_Alt_Module

waves: class
signal_raw: double
signal_baseline: double
t_alt_peaks: double
params: class

set_waves(): void
set_signal(): void
analyze(): void
get_t_peaks(): double
get_params():class

T_Wave_Alt_Params

Params
Setting
Getting