Application Note

OB1203

Pulse Oximeter Algorithm for SpO2, Heart Rate, and Respiration Rate

This application note introduces the OB1203 SpO2 (saturation of peripheral blood oxygen) and heart rate (HR) algorithm including respiration rate (RR) detection. The focus is theory of operation and methods used.

Target Device

Hardware is the OB1203SD-RL-EVK or OB1203SD-RL2-EVK Heart Rate, SpO2, and Respiration Rate Evaluation Kit with the RL78/G13 (R5F100BG) 16-bit microcontroller and OLED Display. Code is in C. Build environment is E2 Studio, with a CC-RL compiler.

Note: The algorithm is provided for reference, demonstration, and/or evaluation. No performance is guaranteed or warranted, nor is fitness or suitability for any medical device application claimed. Renesas will not indemnify customers using any part of the provided reference code. OEM is responsible for performance of products made using any of the provided reference code.

Contents

1. Algorithm Overview

1.1 Algorithm Highlights

The OB1203 example algorithm for heart rate (HR), saturation of peripheral oxygen (SpO2), and respiration rate (RR) features several unique and power signal processing methods implemented for use in small microcontrollers (MCUs) like the Renesas 16-bit RL78 MCUs. Example code for Renesas RA MCUs and Renesas Dialog Bluetooth devices is also available. The algorithm runs in approximately 20ms on a 16-bit MCU.

Features include:

- Efficient (recursive) window-based heart rate (HR) detection
- Robust and simple threshold crossing methods for HR tracking
- Finger pressure detection and correction for oxygen saturation (SpO2)
- Measurement of dynamically-changing breathing (respiration rate)
- Efficient “poor man’s” Savitsky-Golay filter for noise removal
- Kalman filters for outlier removal
- Sensor LED current autogain control
- User access to all algorithm methods and tuning parameters

1.2 Related Content

- For full details on configuring the algorithm options, see the *OB1203 Example Code Application Note*
- Evaluation kits, design guides, samples, and example code are available on the OB1203 product page.

1.3 Main Operation Loop

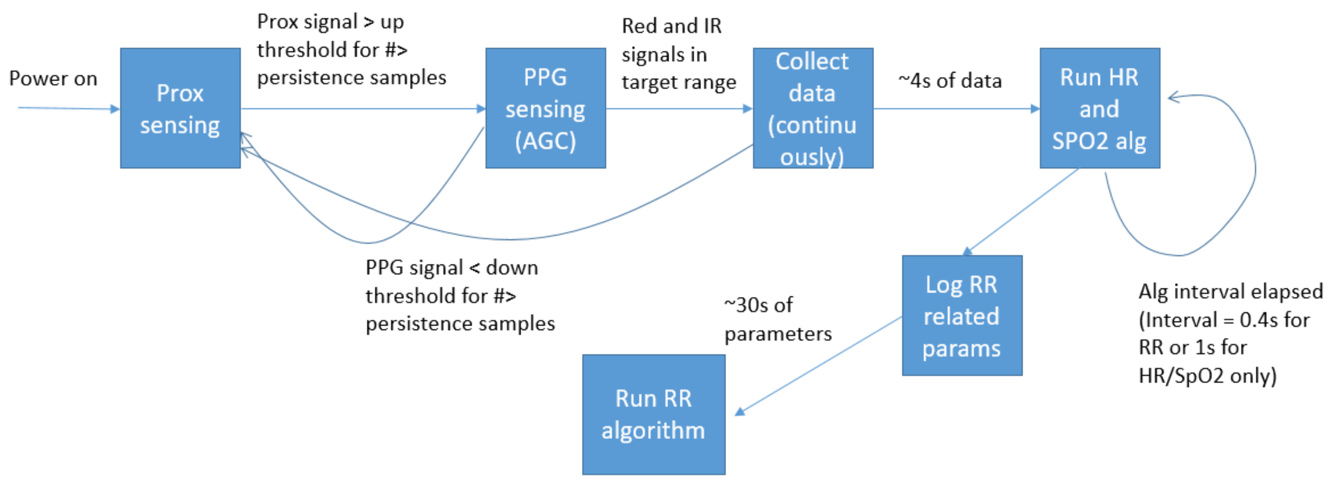
Power on
Prox sensing
Prox signal > up threshold for #> persistence samples
PPG sensing (AGC)
Red and IR signals in target range
Collect data (continuously)
~4s of data
Run HR and SPO2 alg
PPG signal < down threshold for #> persistence samples
Log RR related params
~30s of parameters
Run RR algorithm
Alg interval elapsed (Interval = 0.4s for RR or 1s for HR/SpO2 only)

Figure 1. Main Application Operation

After the start button is pressed, the application starts in a low-power proximity-sensing mode. When the OB1203 detects a finger, it asserts the interrupt pin (active low). The application firmware then switches the OB1203 over to PPG mode (PPG = photoplethysmography, the sensing of blood in tissue with light absorption). An autogain control (AGC) loop adjusts the LED intensities to get within a specified target range of a setpoint. If necessary for dark skin, the sensor gain is increased. The application then begins buffering data from the sensor FIFO RAM. A few seconds of data are buffered before the full algorithm runs. Thereafter, the algorithm runs every `INTERVAL` number of samples.

The algorithm nominally runs on 100 samples per second output data rate, but also includes an option to run at half sample rate (50 samples per second). For more information on algorithm configuration options, see the *OB1203 Evaluation Kit Example Code Application Note* on the Renesas website.

For respiration rate (RR) measurements, `INTERVAL` is 40 (0.4sec). For heart rate (HR) and peripheral oxygen saturation (SpO2) only, `INTERVAL` can be 100 (1sec). From the measured data, HR and SpO2 are estimated and waveform characteristics are measured to calculate respiratory rate (RR). Respiratory rate algorithm runs on about 30 seconds of buffered HR data. It is very sensitive to motion and is only indicated for stationary subjects with good finger stability and/or with a finger clip and no motion.

The sensor is typically covered with a cover glass or surrounded by a finger support to improve stability and data quality. If the signal quality leaves a wider bound, then data collection stops and the application returns to automatic gain control (AGC) mode to re-adjust the LED current levels. If a persistent low level is reached with the LEDs maxed out and sensor gain maxed out, then the application reverts to proximity sensing mode.

In the demonstration code, the following folders/libraries/code blocks are provided:

- SPO2 – This code contains the heart rate, SpO2, and respiratory rate algorithms. The user will call `spo2_init()` before starting measurements and then call `add_sample()` after getting data from the OB1203. Each interval the main application must call `do_algorithm().`
- KALMAN – This code contains an outlier filter routine, referencing an array of structs where each struct contains the parameters of a different filter. There are five filters in the struct array: threshold, heart rate, spo2, respiratory rate, and peak amplitude.
- OB1203 – This code contains the routines needed to set up OB1203 and read data.
- SAVGOL – This code contains a high-efficiency variant of the Savitsky-Golay filter for reducing noise without distorting the waveform
- OXIMETER – `Oximeter.c` is the main program with two support .h files: `oximeter.h` which handles UART and display related things and `oximstruct.h` which is where all the high-level options are configured and the parameters for running the algorithm, sensor status, and AGC control are located.

1.4 Timing

In biosensing mode, an auto gain control (AGC) loop adjusts the LED intensity to achieve a desired signal level, about 80% of full scale. This is necessary to get high signal-to-noise (SNR) from the LED drivers and analog digital converter. Data is read after every ADC conversion result (PPG interrupt). In this mode the LED current is adjusted after each new data. When the infrared and red signal levels are both in range, the device is switched to “FIFO mode” where 10 samples each of red and infrared (IR) are read out from the OB1203 data FIFO RAM every 100ms. Algorithm steps and display updates are done between sample measurements.

The main application runs in `oximeter.c`. It sets up the OB1203 in proximity mode and then waits for an interrupt indicating a finger is present. It then calls `switch_mode()` to change to HR/SpO2 sensing PPG mode. During AGC, a PPG interrupt is used to alert the microcontroller (MCU) after each sample is acquired. When IR and Red channels are both in range, operation switches to “FIFO mode” to read out ten samples from the FIFO each time the FIFO_A_FULL interrupt is asserted. The following sequence, depicted schematically in Figure 2, is followed.

1. Wait for interrupt. Read 10 samples each of IR and red.
2. Call do_algorithm().
3. Wait for interrupt. Read 10 samples each of IR and red.
4. Update the display.
5. Until `INTERVAL` expires: Wait for interrupt. Read 10 samples each of IR and red (i.e., for `INTERVAL` 40, read samples once more).

If PPG wave display is enabled in `oximeter.c`, the PPG waveform portion of the display is updated after the third sample measurement.

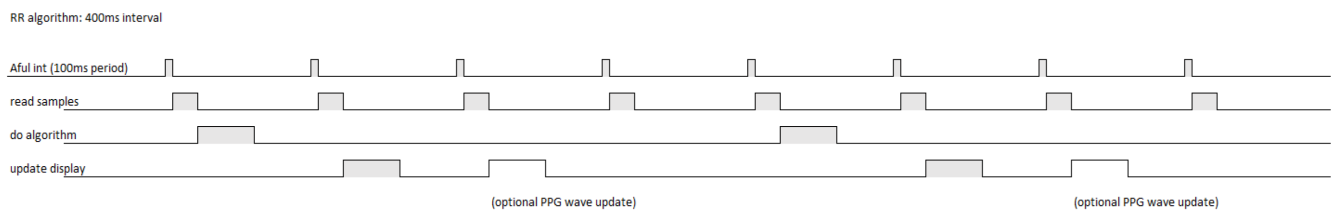
RR algorithm: 400ms interval
Aful int (100ms period)
read samples
do algorithm
update display
(optional PPG wave update)
(optional PPG wave update)

Figure 2. Sample Collection and Algorithm Timing Diagram

Recommended operation parameters for best SNR are 1600Hz sample rate for both IR and Red, with 16 on-chip averages for an output data rate of 100Hz, a setpoint value near 200,000 ADC counts, and FIFO_A_FULL interrupt-based readout of the FIFO. For more information, see Example Code.

1.5 Main Program Settings Configuration

In `oximstruct.h` there are several user-configurable settings that are set with #define lines in the C code. For more information, see the *OB1203 Example Code Application Note*.

1.6 Algorithm Flow

The OB1203 demo algorithm runs two complementary algorithms in parallel, as displayed in the following figure.

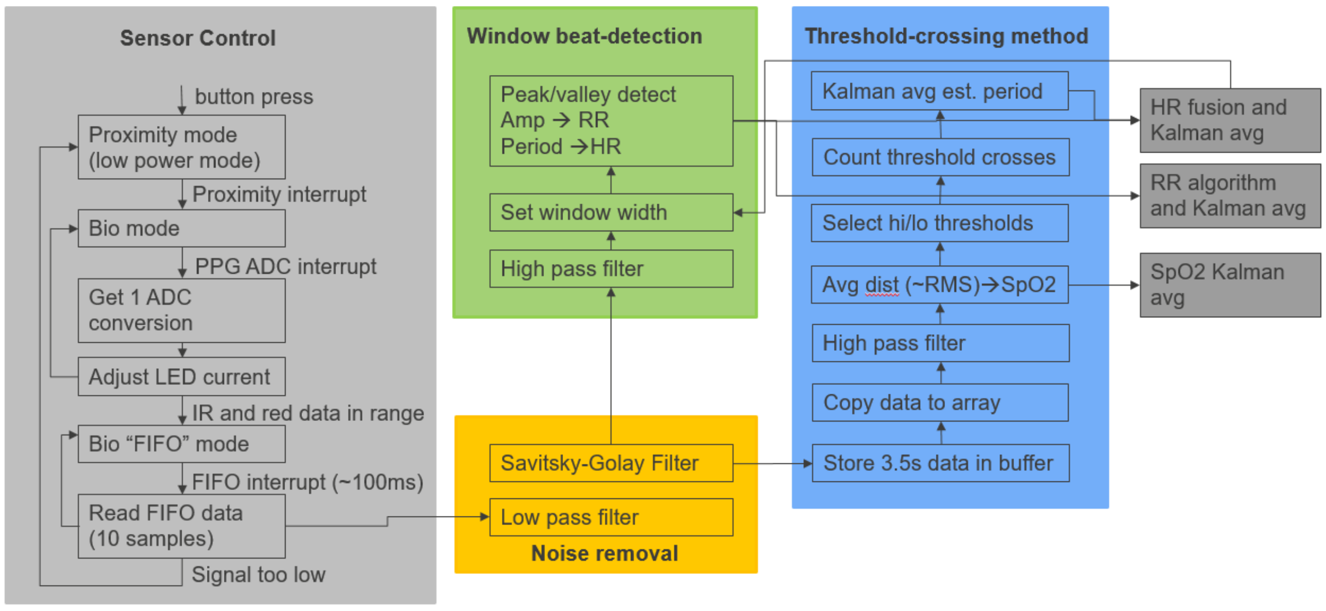

Figure 3. OB1203 Demo High-Level Operation Algorithm Flow

Preprocessing include a moving average low pass filter and modified Savitsky-Golay noise filter.

The filtered data is then passed to two algorithms, a high pass filter, threshold crossing coarse heart rate detection algorithm, and a window-based beat detection method.

The beat-to-beat algorithm has the highest accuracy but is susceptible to second harmonics from detection of dicrotic notches and requires slightly better SNR (perfusion). To verify the accuracy of the respective methods a rough estimate of the heart rate is obtained using a high pass filter and counting threshold crossings. By fusing the two independent measurements of heart rate a consistent heart rate result is obtained.

Note, SpO2 and heart rate results use a zero-order Kalman filter for removing outliers and generating a biologically relevant eight-second average.

The demo code, including display features, requires about 8-9kB of RAM and ~77kB of flash.

2. Algorithm Description

2.1 Preprocessing

In the data preprocessing, a moving average is used as a low pass filter to limit noise bandwidth to biologically relevant frequencies for PPG. A four sample moving average is used for 50 samples per second (sps) date or an eight sample moving average for 100sps data. At high heart rates (>120bpm) the filter length halves and filter length is restored again below 110bpm.

A modified Savitsky-Golay (SG) filter de-noises the signal. An SG filter does a moving curve fit to the data, removing noise *without distorting the signal amplitude*. For smoothly varying signals like PPG, any section of the data can generally be well fit to a quadratic (2nd order SG filter). However, the curve-fit involves a matrix pseudo-inverse that can require order N^3 multiply and divide operations, where N is the number of points to be curve fit in each step. This can be too time consuming for small MCUs. However, if quadratic order is used exclusively, the order N^3 curve fit can be simplified to an exact solution requiring only a few multiplications by a clever switch of operations. Rather than solve for a curve that is best fit to all N points, we calculate an exact fit to three spaced-out points using only two multiply operations. Then we increment those three spaced-out points through the points to be filtered and average the resulting N curve fits at the center point. Then add a new data point, remove the oldest and repeat N curve fits and average for the new center data point. By choosing sample intervals (number of points to be filtered) that are powers of two (e.g., 4, 8, 16), the averaging involves only addition and a bit shift. This “poor man’s” SG filter runs in order N time, a quadratic speedup over published methods. The algorithm runs the SG filter over up to 16 samples at 100sps or 8 samples for 50sps. At high heart rates the filter length is halved to admit the requisite signal bandwidth.

2.2 Heart Rate Algorithm

2.2.1. Threshold-Crossing Method for Heart Rate

The threshold-crossing method for heart rate (like counting zero crossings) counts the number of times the signal crosses a threshold value. Since the PPG signal is asymmetric, we move the threshold slightly away from zero to avoid counting zero crossings from the secondary peaks near dicrotic notches.

From a 3.5 second data buffer of 100 sample per second data, we compute a centered moving average of 64 samples and subtract that from the data to obtain a high-pass-filtered result. Figure 4 shows the effect of the high pass filter on low heart and high heart rate data. The effect is similar to a derivative and disambiguates the pulsile waveform from baseline drift or motion. Notably, this method avoids detecting oscillations in the PPG signal following the systole (dicrotic notches) as heartbeats. Note, the data is raw, not inverted as is typically done with PPG sensor displays, so the systole shows up as a valley in this data.

The threshold is chosen as 1/3-1/2 (user adjustable) of the max PPG signal in the PPG data buffer.

With the high pass filtered data, we measure the average distance of the data from zero. This is an efficient, RMS-like estimate of the signal’s AC amplitude, which is used in the calculation of SpO2.

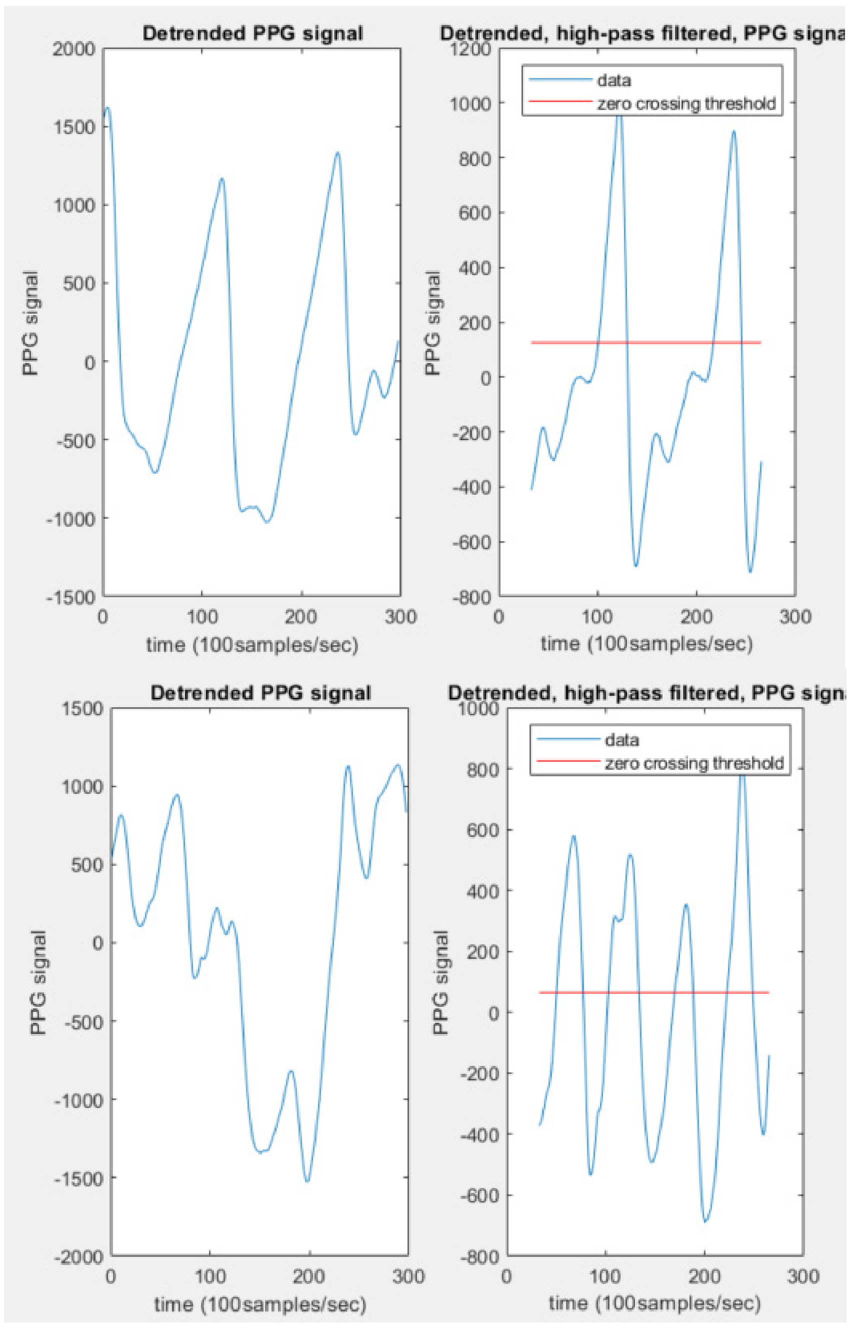

Figure 4. Threshold Crossing Method

In practice, too much finger pressure can distort the waveform and the dicrotic notch can appear elevated in the signal and cross the threshold, resulting in overestimated heart rate. To avoid this, we use two thresholds: one positive and one negative and choose the lower of the two threshold crossing counts. Additionally we use a "de-bouncing" feature for heart rates below 140bpm, wherein any threshold crossing within less than half of the current estimated heart rate from the previous crossing in the same direction is not counted.

2.2.2. Window Method for Heart Rate

True beat detection can be challenging for PPG waveforms due to a secondary peak following the dicrotic notch feature. Complex logic comparing peak heights, time since the last peak, etc., can be used to attempt to find the correct peaks, or tracking filters can attempt to smooth out the secondary peaks.

However, our method uses an extremely simple, extremely fast and very robust "moving window" method for detection of true peaks (diastoles) and valleys (systoles) in the incoming sensor data. (Note, in hospital equipment the PPG waveform is inverted so systoles show up as peaks as this correlates to pressure in the arteries.) The window method is run each measurement interval (e.g., 40 samples every 400ms). Steps are as follows:

1. Set the window width to 2/3 of the current estimated heartbeat period. Initially, the estimated period is determined based on the threshold crossing method. Subsequently, heart rate period is determined by a fusion of the window and threshold crossing estimates. Correct window width selection is *critical* to detecting true peaks and no false peaks with the window method.
2. For each new sample, check if it is a larger than the previous peak or smaller than the previous valley. If a sample remains a peak or a valley for the entire window width it is a true peak or valley.
3. Properly handle special cases like window width changes.

Note, this method does not require a buffer of the samples in the window, nor does it require comparing the new sample with every sample in the window. It is sufficient to log the positions and values of the candidate peak and valley and compare with the latest sample. As a result, this method is both extremely fast and extremely low on system resources.

Visually, if we display the window maximum and minimum the number will appear to “freeze” when we hit a peak or valley and then release when the window moves off the peak or valley. If the value freezes for the entire window width, it is a true peak or valley.

The reason this method works is that the dicrotic notch is within one window width of the main systole feature (valley in raw data, peak in hospital equipment displays). Therefore, the secondary peak is not the extreme value for the entire passage of the window across the feature, and it does not qualify as a true extremum (i.e., it does not “freeze” long enough). The peak-valley difference, sometimes called pleth variability index (PVI), is logged for use in respiration rate detection.

Note, noise removal by the moving average low pass filter and SG filter is beneficial to the accuracy of the peaks and valleys.

2.2.3. Fusion Method

This window method an example of a tracking filter. In practice, when it is “on” the window method is always more accurate than the threshold crossing method. However, by itself, the window method offers no guarantee that the window is set correctly. If set initially based on an incorrect estimate or assumption of the heartbeat period, this method can also lock into a subharmonic, or second harmonic. So the algorithm needs a way to “break out” when this occurs.

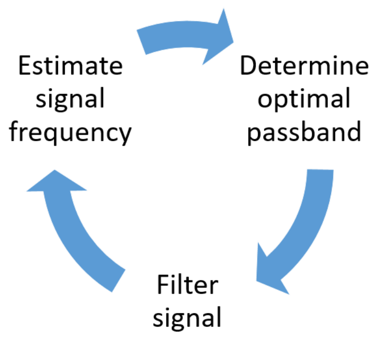

Figure 5. Tracking Filter Principle

The solution implemented in the demo is to fuse multiple algorithms with complementary benefits. The threshold-crossing method, while less accurate, makes no assumptions about the signal periodicity. So, we absorb some minor decrease in accuracy in rare cases where the threshold crossing is fooled by heavy breathing or moving, in exchange for the ability to “break out” and avoid gross errors.

Additionally, a zero-order Kalman (outlier) filters will reset the algorithm if too many outliers occur in a row, allowing the filter to more rapidly re-acquire and track the correct frequency.

The heartrate fusion method in the function `calc_hr()` combines estimates of heart rate from the threshold-crossing method and window method.

Fusion logic is as follows:

1. If the window method does not produce a heart beat, use the threshold-crossing estimate.
2. If the window method is more than 25% from the threshold-crossing estimate, ignore the window method result. Note, errors tend to be missing a beat, such as an ectopic beat (skipped beat) or counting a dicrotic notch during motion, so if the measurement is off, the error is most often on the order of 100%. When those extraneous detections are removed the variance of either method drops to less than 25%.

2.3 SpO2 Algorithm

Reflective PPG is measured against the skin with emitter and receiver on the same side, similar to an active infrared (IR) proximity sensor. Some PPG signal derives from the change in blood volume in the tissue, increasing absorption as the vessels upstream of the capillaries fill after the heartbeat. Another portion of the signal derives from agglomeration of cells at low blood flow rates and disaggregation at higher flow rates.

Reflective PPG measurement, while convenient in mobile devices, is not optimal for individuals with very soft fingers including young children and some elderly individuals.

The observed signal has a large static or DC component on which the small AC cardiovascular signal appears. The ratio of AC/DC is called perfusion index (PI).

$$PI = \frac{IR_{AC}}{IR_{DC}} * 100\%$$

Perfusion index increases when there is more blood “perfused” into the tissue. That actual number used for PI depends on several details, including wavelength and LED/receiver configuration. PI is important because instruments and algorithms have a lower limit for perfusion index below which the SpO2 calculation loses accuracy.

For transmissive systems shining light through the finger, transmission perfusion index (tPI) is on the order of several percent, meaning the AC ripple is about 10 to 100 times smaller than the DC signal.

Reflective systems measure PPG with the light source and detector from the same side of the finger. With a smaller sample region and shorter effective absorption path length, less modulation of the PPG from the pulsing blood observed. Thus, reflective perfusion index (rPI) is necessarily smaller than tPI.

Note that reflective and transmissive PI are not physically equivalent since different tissue regions are measured. Even so, the tPI and rPI perfusion measurements for transmission and reflective devices are useful in evaluating the validity of the respective SpO2 measurements.

Note, in the example algorithm, rPI is estimated from the infrared signal “RMS” or average deviation and the mean DC count level. The displayed rPI value (`avg_PI`) is given in equivalent peak-to-peak percentage of DC level and includes a ~5 second moving average.

2.3.1. SpO2 Mechanism

The measurement of the saturation of blood by oxygen is accomplished with PPG measurements at two wavelengths. The most common wavelengths are red and IR, due to the strong differential absorption, availability of light sources, relatively flat spectra (for wavelength shift tolerance). Essentially, the red channel absorbance changes with oxygenation, but the IR does not, so we can use the IR as a stable reference level.

The blood oxygen saturation level or SpO2 level is calculated from a calibrate curve plotting SpO2 versus ratio red and IR signals. This is done by calibrating against a known instrument or blood test reference in a hypoxia lab using breathe-down testing. This calibration plot, also known as the “R curve,” depends on the mechanical details of a PPG system.

The ratio of ratios “R” is given by:

$$R = \frac{R_{AC}/R_{DC}}{IR_{AC}/IR_{DC}}$$

SpO2 is calculated with positive coefficients (in a linear approximation) as:

$$Spo2 = (c_0 - c_1 R) * 100\%$$

Higher order terms or a look-up table can be used. Physiologically, less oxygen means a larger red AC signal.

The R value or “ratio of ratios” is effective in cancelling out DC variations from finger placement, skin tone, etc., while preserving changes specific to blood chemistry in the ratio of the pulsile (AC) signals.

If anything in the mechanical or optical design changes (e.g., different enclosure color, shape, or materials), or even the pressure of the finger clip, the hypoxia test calibration procedure must be redone to obtain a new R curve.

SpO2 requires a much higher SNR than heart rate because we are not merely trying to detect the heartbeat which is a small ripple on a big background, we are trying to quantify the size of that ripple and divide it by another very small ripple. Tight control of red LED wavelength, or wavelength-specific calibration is important. In addition, artifacts due to finger motion should be identified and removed from the data or compensated.

The AC signal can be estimated from peak-to-valley distances; however, this requires low noise data. An alternative method that is robust to moderate levels of noise, but susceptible to rapid DC changes, is to calculate the root mean square (RMS) deviation, also known as standard deviation, of the signal. It also has the advantage of being numerically robust to digitization noise in smaller amplitude signals. In our case, we compute the average distance from zero as it is more numerically efficient, though conceptually similar to RMS.

This AC signal amplitude estimate can be obtained either from the high-pass filtered data in the recursive window method, or from the high-pass filtered data in the threshold crossing method. In theory they are identical; however, a fully recursive method is more susceptible to data errors that continue to propagate, whereas the threshold crossing method amplitude is calculated ab-initio from the buffer each time, so any unexpected errors do not propagate. Therefore, the signal amplitude from the threshold crossing method is preferred.

For the SpO2 algorithm, a special asymmetrical Kalman filter is used which allows higher jumps up than down. This is because the human body oxygenation can return to normal in a single breath after hypoxia, whereas decreasing SpO2 requires a much longer time.

For very low perfusion index, estimated SpO2 values will be slightly lower than expected, due to the effect of noise or drift affecting the smaller red signal more than the infrared signal, the R value being therefore higher than expected and the resulting SpO2 value being lower.

2.3.2. Correction of Finger Pressure

SpO2 device calibration depends on the finger pressure. This effect is larger in reflective PPG devices, which have a smaller sensing region. When a finger clip mechanical design is less than optimal a large finger experiences more pressure on the sensor than a smaller finger. Other use cases may rely on the user providing the correct pressure.

Typically, higher pressure results in the SpO2 estimate being lower than expected. Higher pressure is visible in the raw PPG waveform data as a “squaring off” of the waveform in the diastole region. It begins to look more trapezoidal. If this effect can be quantized, the finger over pressure can be estimated and corrected.

Ostensibly, characterizing the waveform shape this would require some form of time-frequency analysis to characterize harmonic content or machine learning, however, there is a simpler way. When a moving average baseline is removed from the PPG waveform, especially if the width of the moving average is less than the beat period, the effect of the finger pressure becomes evident in the peak-to-valley ratio.

The OB1203 algorithm includes an optional correction method for finger over pressure based on the peak-to-valley ratio.

When enabled, the finger pressure correction applies a linear compensation to the reported SpO2 value for valley-to-peak amplitude ratios larger than a minimum.

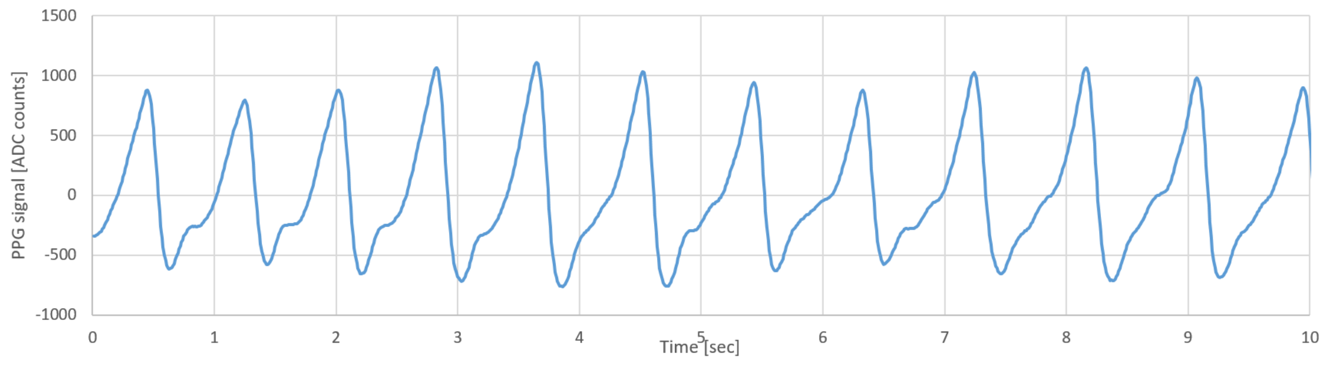

Figure 6. Low Finger Pressure PPG Waveform after High Pass Filtering (Baseline Removal) showing Diastole Peaks with Larger Amplitude than Systole Valleys

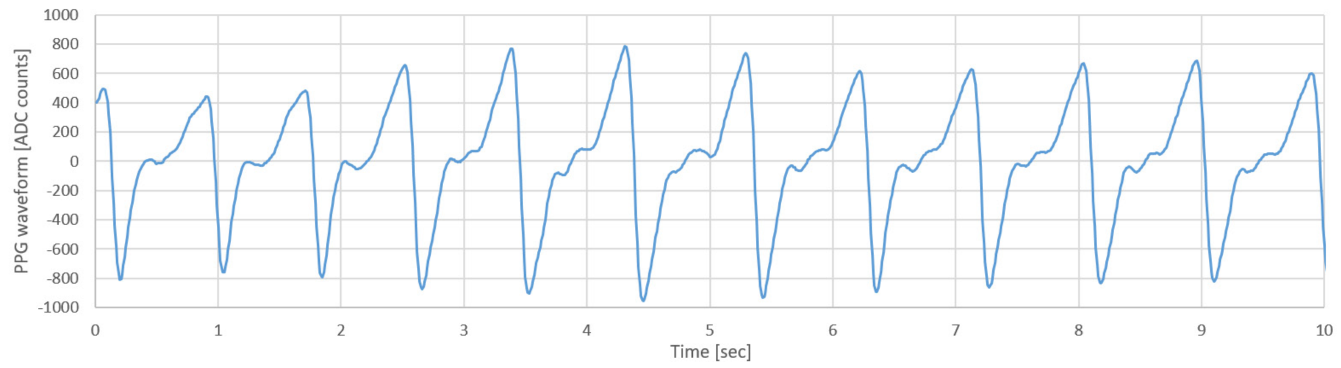

Figure 7. High Finger Pressure PPG Waveform after High Pass Filtering (Baseline Removal) showing increase in Relatively Larger Systole Valley Amplitudes

2.3.3. Noise Compensation

Measurement noise affects SpO2 accuracy at low perfusion levels. Noise adds to the signal in quadrature.

$$Meas^2 = Signal^2 + Noise^2$$

When signal is much larger than noise, the effect of noise on the measurement is negligible. However, as the signal approaches the noise floor, the noise dominates the measurement amplitude or RMS. In SpO2 measurements the red channel signal is typically about half the size of the infrared channel. As a result, the red channel becomes dominated by noise sooner. Its value no longer decreases in proportion to the signal.

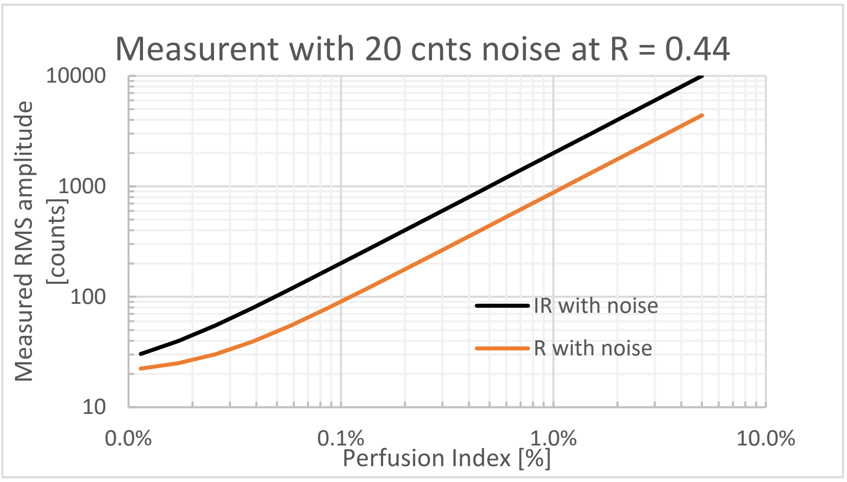

Figure 8. Effect of Noise on Measurement RMS Amplitude is Prominent at Low Perfusion for the Smaller Red Channel Signal

SpO2 is inversely proportional to the red to infrared ratio, so in the presence of noise, the estimated SpO2 value is lower than expected.

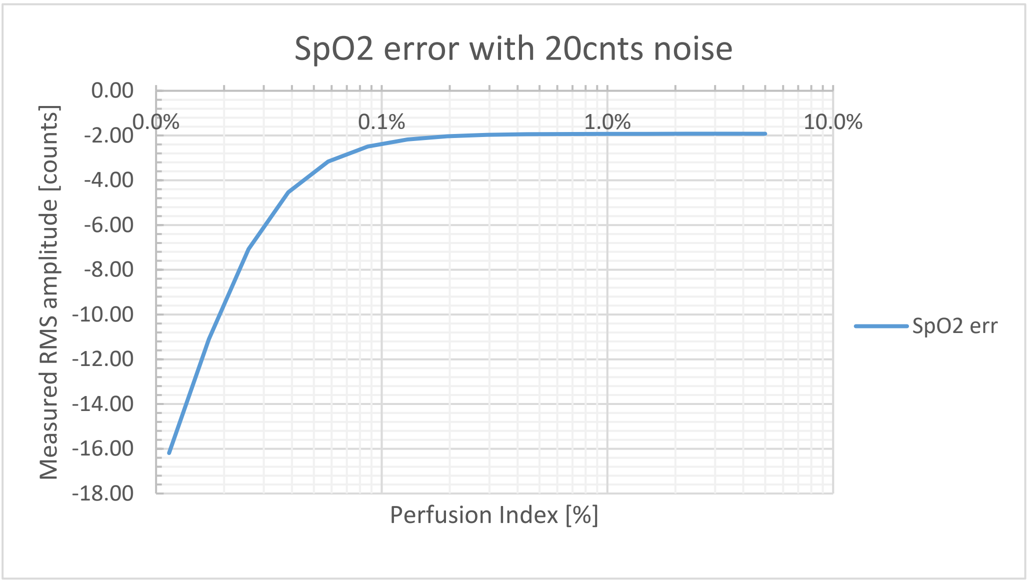

Figure 9. Spo2 Error Increases at Lower PI Values in the Presence of Noise

The measurement noise is primarily scales with LED intensity when the limiting source of noise is transmit (Tx) noise. This is the case for LED currents greater than about 50mA and count values above about 60,000 (with ADC gain 1). Thus, the measurement noise for each channel will be proportional to the LED current.

The algorithm can partially compensate for the effect of measurement noise on SpO2 by subtracting the estimated noise from the measurement.

$$Estimated\ signal = \sqrt{Meas^2 - estimated\ noise^2}$$

This can reduce the effect of noise on SpO2 measurements at low perfusion.

We make sure measurements and noise are equivalent, for instance both signal and noise should be estimated the same way (e.g., RMS, average deviation, or peak to peak). In the algorithm, average deviation is used.

To measure signal noise, a target card is placed over the sensor and raw data is exported. To measure noise in the relevant bandwidth we apply a moving eight sample average and remove a 64 sample centered moving baseline, identical to the way the signal is processed in the SpO2 algorithm. We then calculate the average deviation. The measured noise will vary from device to device based on the LED trim so for any hardware design the noise estimate will not be perfect and variance will increase (noise is random). For customer applications, a lower bound should be set on the perfusion index below which SpO2 values are not be reported, typically in the range of 0.05% to 0.1%, depending on the achieved noise performance (see *OB1203 RL78 Example Code Application Note* “CHECK_PI” option).

2.4 Respiratory Rate Algorithm

Respiratory rate can be measured by the modulation of the heartbeat according to the amount of air in the lungs, creating the “sinus rhythm” in the heartrate. When the lungs are full, baroreceptors on the heart cause it to take more frequent heart beats with lower stroke volume. The reduced stroke volume increases the DC level of the PPG signal. This is called respiration induced “baseline wander” or intensity variation (IV). Likewise, the changing stroke volume modulates the PPG amplitude (AM), while the changing heart rate produces a frequency modulation (FM), as displayed in Figure 10.

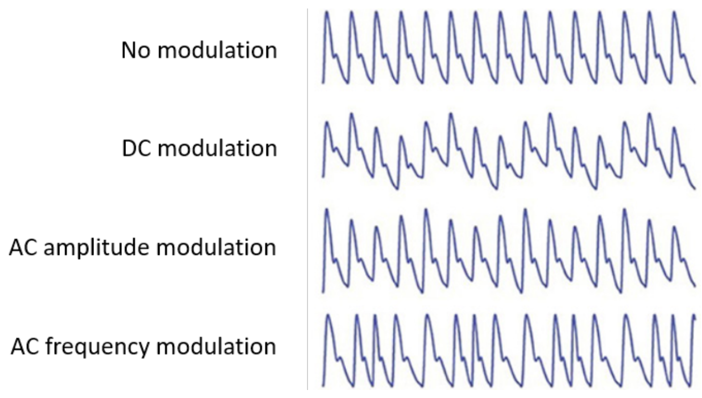

Figure 10. Modulation of PPG by Respiration

Tracking the systole peak value (valley in raw data) or peak-valley amplitude encodes amplitude modulation and is a robust method of tracking respiration. This amplitude modulation is called pleth variability and the size of the modulation is called pleth variability index or PVI.

This is similar to demodulation of audio from an AM radio signal. The heartbeats are the carrier wave. The slowly varying envelope (PVI) is due to the breathing.

For respiration detection, the envelope amplitude estimate comes from the peak-valley differences measured in the window method. Once we have a buffer of beat amplitude values, we look for periodic changes. Essentially, we are detecting a tiny slow ripple (breathing) on top of tiny ripple (heartbeat) in the PPG signal. We also have the challenges of breathing being irregular, plus more random drift and extraneous autonomonic (subconscious biological) effects. In short, it is a lot harder than detecting heartbeats.

For example, if we tried to use the more easily detected frequency variation, we find that FM modulation is also subject to autonomic variations unrelated to breathing. It can also have difficulty following fast breathing. For this reason, we prefer PVI (pulse amplitude). Requirements on the accuracy of the PVI are demanding, so this measurement does not accommodate user motion.

Additionally, due to random autonomic variations, it is typically not possible to distinguish “no breathing” from “slow breathing”, so PPG-based respiration measurements will have a lower limit on respiration rate around 5rpm (respirations per minute – not to be confused with revolutions per minute in car engines).

Since we log PVI values at a fixed rate (INTERVAL/SAMPLE_RATE) and heart rate is not commensurate, we get significant “jitter” in addition to signal wander noise from physical changes. Therefore, peak counting methods reliant on a periodic signal are somewhat troublesome for RR determination. For example, a fast Fourier transform (FFT) typically generates multiple possible respiration peaks and choosing the right peak will require some knowledge about the signal history. More advanced methods include wavelet transforms, synchro squeezing, and auto-regression.

However, for our purposes a zero-crossing count method convenient for 16-bit processors proves to be robust – *provided a suitable baseline removal filter is used*. This sets up a “chicken and egg” problem, where lack of information about RR means we cannot select the right filter for detecting RR. Some of the key methods in the demonstrated RR detection algorithm relate to how the filters are adjusted in order to (1) acquire quickly, (2) adjust to changes, and (3) provide reasonable signal stability. Note, not all commercial device algorithms can adjust to sudden changes in RR.

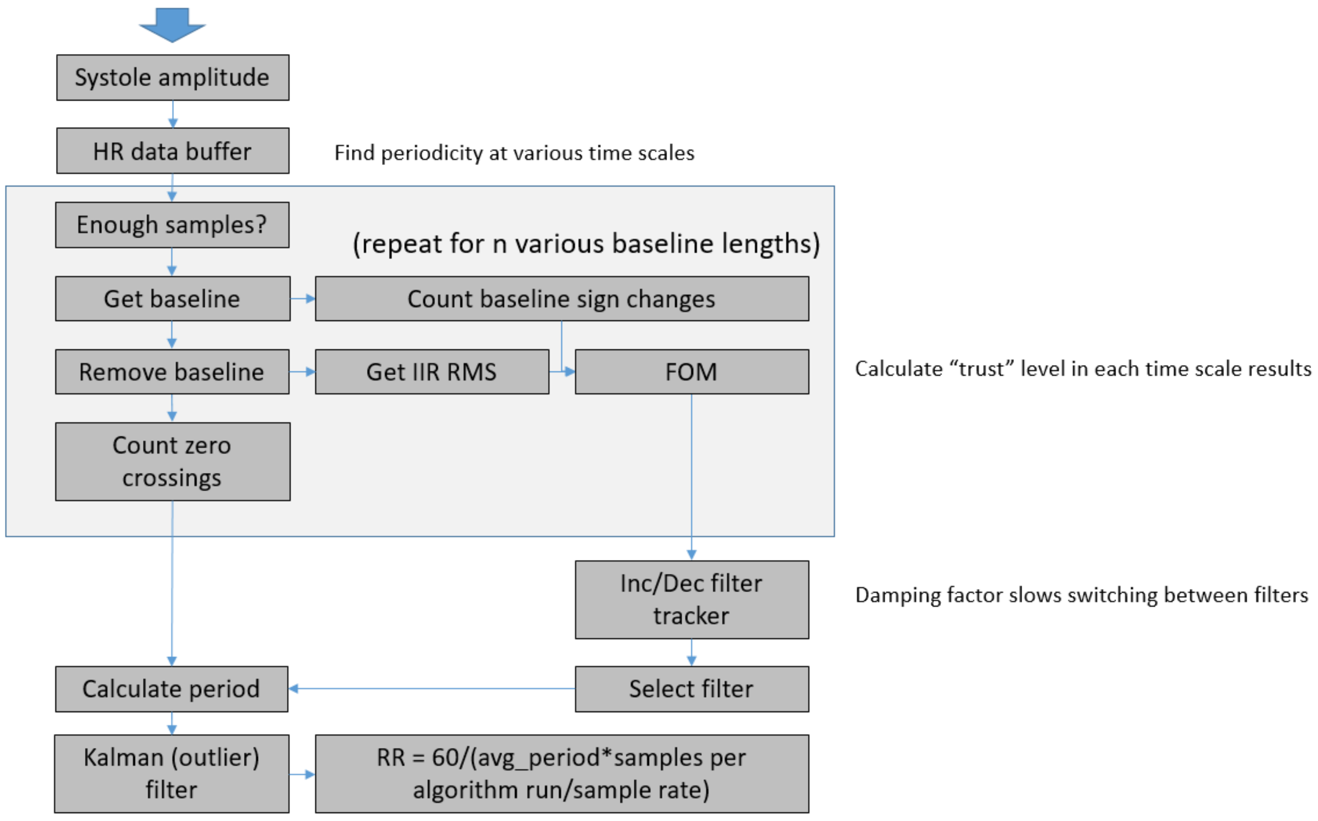
Systole amplitude
HR data buffer
Find periodicity at various time scales
Enough samples?
(repeat for n various baseline lengths)
Get baseline
Count baseline sign changes
Remove baseline
Get IIR RMS
FOM
Calculate "trust" level in each time scale results
Count zero crossings
Inc/Dec filter tracker
Damping factor slows switching between filters
Calculate period
Select filter
Kalman (outlier) filter
RR = 60/(avg_period*samples per algorithm run/sample rate)

Figure 11. Respiration Rate Algorithm Flow

The critical parameter in a zero crossing method is the length of the baseline used to level or de-trend the data. Since the respiration rate is not known a-priori, an appropriate length filter can't be chosen.

Since zero crossing continuing is so fast and easy we can calculate zero crossings for a variety of possible baseline lengths ranging from short values ideal for detecting rapid breathing to long baseline values ideal for detecting slow breathing. To choose the right filter we calculate a figure of merit based on the ratio of the stability of the resulting RR estimates (tends to favor slower breathing rates) and the number of sign changes in the baseline (tends to favor higher breathing rates). For stability, a "damping factor" limits the speed with which we allow changes in the baseline filter length.

The application software runs a variable-length moving average on the calculated respiratory rate to improve stability of output. It can be tuned larger or smaller for more stability or less lag.

Due to the nature of respiratory rate sampling from the peak values, we cannot accurately detect respirations faster than twice the heart rate. This is similar to a Nyquist criterion. Based on our limited algorithm run rate, respirations above 30rpm may be underestimated, although in the case of very high heart rate respirations up to 40rpm can be observed. The minimum heart rate is 5rpm.

Important: The RR algorithm requires very low motion. Likewise, sudden changes in breathing amplitude or frequency will temporarily reduce accuracy.

RR data displays about 30 seconds after the heart rate data is displayed and reaches nominal accuracy after about 40 to 50 seconds when signal statistics are available.

3. OB1203 Algorithm Walkthrough

The main program fetches data from the OB1203 at 100Hz for both red and IR in function `get_sensor_data()`. It arranges the bytes from the FIFO and calls `add_sample()`. The function `add_sample()` keeps a running average of eight samples (12Hz) and loads the filtered data into a circular buffer. The buffer is long enough for two beats at the slowest heart rate of ~40Hz, plus an extra sample to make the sum of squares easy, plus the maximum length of the moving average (dicrotic notch) filter. These values are defined in spo2.h.

The algorithm is run once each measurement interval (e.g., 1 second for HR/Spo2 or 0.4 seconds for RR).

The HR and SpO2 algorithm in `do_algorithm()` are in two pieces: `do_algorithm_part1()`and `do_algorithm_part2()` (threshold-crossing). New samples are read from the FIFO between running the two portions of the algorithm. `do_algorithm_part3()` and `do_algorithm_part4()` do the respiratory rate (RR) algorithm and run immediately after part 2.

`do_algorithm_part1()` copies data using `copy_data()` from the circular buffer into linear arrays `AC1f`, obtains the DC value for PI and R calculations.

`do_algorithm_part2()` performs the high pass filtering of the data (`high_pass_filter()`), window detection or heart rate and coarse estimate of heart rate using the threshold crossing method. It also calculates PI, and SpO2 using both methods, performs the biological average (8 seconds) via respective Kalman filters, and arbitrates which method's result to use for the display out.

`calc_hr()`, `calc_R()`, `calc_R_p2p()`, and `calc_spo2()` `are` self-explanatory. The R value calculated in `calc_R()` is based on the average distance from zero (quasi-RMS). The R value from `calc_R_p2p()` is the peak-to-valley difference.

The spo2 calculation in `calc_spo2()` uses a polynomial in R to calculate the SpO2 level from the most stable R value.

In `do_algorithm_part2()`, the following steps are performed:

1. Run the recursive window method for heart beat detection `hpf_window_peak_detect()` and the optional pressure detection method `get_pressure()`.
2. High pass filter the data using `high_pass_filter()`. A centered, moving 64 sample (0.64 sec) average is subtracted from the data in the buffer. This is necessary to flatten out the oscillations from motion and breathing.
3. Calculate the positive threshold as 1/3 to 1/2 of the maximum signal in the filtered data and the negative threshold as 1/3 to 1/2 of the minimum signal.
4. Count threshold crossings and convert to an estimated HR period using 60 sec/min * 100 samples/sec / (X/2), where X is the number of threshold crossing (i.e., two per beat). This is retained in fixed point precision.
5. Run a Kalman filter to find and remove outliers in threshold crossings.
6. Negotiate the best HR value in function `calc_hr()`.
7. Calculate the ratio of ratios `calc_R()` with optional pressure correction.
8. Calculate spo2 `calc_spo2()`.
9. Run a Kalman filter to find and remove outliers in both SpO2 and HR and perform an 8 second (clinical) moving average.

The algorithm then does `do_algorithm_part3()`. This is where the algorithm buffers the latest heart beat amplitude and calculates the approximate respiration rate (RR). RR is calculated by constructing a variety of moving averages or baselines. The algorithm calculates the number of zero crossings relative to each baseline over a length of samples `MAX_BREATH_FILTER_SPAN` is equal to 1.6 times the longest moving average. This is intended for accuracy for slow breathing. `MAX_BREATH_FILTER_SPAN` can be optimized for more accuracy or faster startup.

We characterize the baselines by counting the number of sign changes of the baseline. More sign changes mean a better baseline. It is believed that the flatter a baseline is (i.e., it spans an integer number of breaths), the more sign changes occur, since small fluctuations can cause a sign change on a flat signal. This method is not described in literature and may be unique to our algorithm.

In `do_algorithm_part4()` we attempt to choose the best estimate of the breathing rate. We characterize the recent zero crossing calculations for each length of baseline using an IIR filter based approximation of the root mean square (RMS) that is efficient for 16-bit processors.

We calculate a figure of merit (badness) based on the ratio of RMS zero crossings to baseline sign changes. We find the filter with the best figure of merit (lowest badness) and increment the filter tracker towards that filter. A `filter_damping` factor is used in the filter tracker to make it so that the filter switching takes place slowly. This enhances stability. Increasing filter damping means it can take longer to reach the correct value or adapt to a change, but there is less deviation from a correct value once we have it.

The RR value is calculated from zero crossings as breaths per minute, then the value is Kalman-filtered and displayed. An additional configurable duration moving average that can be increased for signal stability or decreased for improved response time.

3.1 Kalman Filters

Zero-order Kalman filters are used for the coarse estimate filter result, the 8-second heart rate average, the 8-second SpO2 average, the respiratory rate, the beat-to-beat period, and the beat-to-beat amplitude.

The Kalman filter code is in `Kalman.c`.

The zero-order Kalman filter checks the RMS variation in the recent data buffer and removes outliers before adding valid samples to the Kalman buffer. The Kalman buffer is used to calculate the average.

Kalman filter parameters are defined in `Kalman.h`. Kalman filters are reset after a defined number of zero inputs (algorithm fails) or too many outliers in a row to allow the filter to more quickly recapture the correct value.

3.2 Data Quality Checks

For health related devices, it is often preferable to not display any value rather than display a potentially incorrect value. Configuration options in `oximstruct.h` allow additional data quality checks enabled by the settings in step 6 by defining `CHECK_PI`. The following conditions result in a data “freeze” and eventual reset of the algorithm:

- PI below minimum threshold
- PI below low threshold and RMS variation in PI greater than the low PI RMS threshold
- PI above low threshold and RMS variation in PI greater the normal PI RMS threshold

4. Example Code

Download the demo source code from the OB1203SD-RL-EVK product page. Contact your helpful local sales representative for assistance.

5. Revision History

Revision	Date	Description
3.01	Apr 25, 2022	Updated the first sentence in “Main Program Settings Configuration”.
3.00	Jan 14, 2022	Major algorithm update including the window method and pressure correction. Removed the triangle FIR filter method.
2.00	Oct 21, 2021	Major algorithm update including the Savitsky-Golay filter and high pass filter methods. Removed the correlation method.
1.00	Jul 23, 2021	Initial release.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks