OB1203SD-BT2-EVK Dialog DA14531 Bluetooth Evaluation Kit Example Code

This document discusses the OB1203 example code used in the OB1203SD-BT2-EVK evaluation kit (EVK). It also describes how to configure options for evaluation and how to debug for custom development.

Contents

1. Notice 1
2. Resources 1
3. Device Specification 2
4. Opening the Project in Keil IDE 2
5. UART Output 2
6. Configuring Settings 3
6.1 Step 1: Compute Respiration or Heart Rate and SpO2 Only 3
6.2 Step 2: Define Power vs. SNR Option 3
6.3 Step 3: Enable/Disable SpO2 Debugging 3
7. Code Organization 4
8. Compiling and Debugging Code 4
9. Loading code 4
9.1 Record Data 4
10. Algorithm Parameters and Tuning 4
10.1 SpO2 Calibration Parameters 4
10.1.1. R-curve 4
10.1.2. Finger Pressure Correction 5
10.1.3. Noise Correction 5
10.1.4. SG Filter Tuning 6
11. Revision History 7

1. Notice

The source code is provided for reference, demonstration, and/or evaluation. Note the disclaimer in the source code `oximeter.c` file header. No performance is guaranteed or warranted, and fitness or suitability is not claimed for any medical device application. Renesas will not indemnify customers using any part of the provided reference code. The OEM is responsible for performance of products made using any of the provided reference code. For general questions, contact Renesas. In addition, for medical devices Renesas strongly recommends to consult a medical device regulatory compliance advisor.

2. Resources

- Renesas Dialog DA14531 microcontroller source code, application notes, datasheets, and resources are located at renesas.com/ob1203. Submit a request for the source code via the website download link, and then return to same link to download the code once the request has been approved.

- Reference code for other microcontrollers such as Renesas ARM-based RA devices and 16-bit RL78 devices are also available. Contact Renesas and/or check the OB1203 website for updates and new releases.
- The evaluation kit hardware is part number OB1203SD-BT2-EVK.
- Custom boards developed by Renesas customers should follow the electrical and optical design guide, with particular attention paid to reduction of noise and EMI on VDD and heat sinking at the LED supply pins.
- The firmware integrated development environment (IDE) is Keil.
- A Dialog flash programming tool such as is available in the Dialog evaluation kit DA14585IOTMSENSKT is required to program the microcontrollers. It can be purchased from Renesas distributors.
- The evaluation kit is shipped preprogrammed.
- Detailed explanations of the Spo2, heart rate, and respiration rate algorithms are provided in a separate application note.
- For more detailed information or questions regarding this product, contact Renesas. We have offices and sales teams around the world to support you.

3. Device Specification

This software is provided for the OB1203SD-BT2-EVK evaluation kit. For more detailed information about the hardware, its components, and capabilities, see the EVK user manual.

Tips:

- The EVK is designed to run from two AA or AAA batteries.
- For best operation, a finger cradle or support is recommended to stabilize the finger on the sensor. These can be 3D printed. Contact Renesas sales if you need a replacement or 3D file for the finger support.
  Note: Should you decide to remove the finger cradle, please lift straight up and pay special attention to sheer forces and avoiding damages to the OB1203 sensor module.

4. Opening the Project in Keil IDE

1. Download the SDK6.0.14.1114 for DA14531 from the Dialog Semiconductor website.
2. Extract the zip file to C:\DiaSemi\ or similar location.
3. Download the Bluetooth EVK source code from the Renesas OB1203 website.
4. Extract the zip file into the following SDK folder:
   C:\DiaSemi\DA145xx_SDK\6.0.14.1144\projects\target_apps\ble_examples\ble_app_plx.
5. Open the Keil IDE.
6. **Project** > **Open Project**:

   C:\DiaSemi\DA145xx_SDK\6.0.14.1144\projects\target_apps\ble_examples\ble_app_plx\Keil_5\
   ble_app_plx.uvprojx

5. UART Output

Raw data and Algorithm results are printed on the serial UART pin at 115200 baud. Select the COM port that shows up when you plug the USB micro port into the PC. Data can be live-view graphed and saved with the free program SerialPlot, or view as scrolling text print lines with a free serial terminal program like TeraTerm or Putty.

The raw data output is particularly useful for looking at sensor noise. For example, you can place a stationary (heavy) target over the sensor to check the noise level or you can view data live using SerialPlot and see how finger pressure affects the signal or log raw data from a subject for later analysis.

6. Configuring Settings

6.1 Step 1: Compute Respiration or Heart Rate and SpO2 Only

In step 1, you can change the rate at which the algorithm runs. For SpO2 and HR only applications the algorithm can be run only once per second. **Note**: More verification was done for the SpO2 option. The setting is determined by whether COMPUTE_RR (respiration rate) is defined in SpO2.h. By default, COMPUTE_RR is enabled and the algorithm runs (40ms) every 400ms. The algorithm runs on fewer samples when it runs more frequently in this mode so the overall difference is only about 20% less computation for not computing RR.

6.2 Step 2: Define Power vs. SNR Option

In step 2, you can optimize for lower power or signal to noise ratio. This demonstrates how a developer might attempt to optimize the operation for a particular signal range. In this step, you set how frequently OB1203 samples. With the default MID_POWER configuration OB1203 oversamples 8x and outputs the averaged (binned) data 100 times a second. The oversampling can be set to 16x by defining BEST_SNR or 4x by defining LOW_POWER.

For the OB1203SD-BT2-EVK the relative signal to noise ratios for BEST_SNR, MID_POWER, and LOW_POWER configurations are 1.4, 1.2, and 1 respectively, with RMS noise in the bandwidth of interest for normal heart rate range and mid-range LED of approximately 10, 12, and 14 counts.

6.3 Step 3: Enable/Disable SpO2 Debugging

Normally, SpO2 algorithm output is clipped to a maximum of 100%, regardless of the calculated ratio of ratios “R” value. In order to calibrate the SpO2 for a particular device, it is advisable to define “SPO2_DEBUG” to avoid clipping data at the upper or lower extremes during hypoxia tests.

7. Code Organization

The code consists of SDK code for configuring and controlling the Bluetooth and application-specific code. Application-specific code is in the folder `user_app`.

`Oximeter.c` is the main program and has two support files `oximeter.h` which has the things related to UART serial print and display, and `oximstruct.h` which has things related to running the algorithm.

`OB1203.c` and `OB1203.h` provide code for configuring and reading out data from the OB1203 bio sensor ASIC.

`KALMAN.c` and `KALMAN.h` provide code for doing outlier filtering and averaging of data. This code is a zero-order Kalman filter where only the variance in the data is analyzed to determine outliers and the predictor is based on a moving average. `KALMAN.h` defines parameters that set the duration of the averaging for displayed SpO2 and HR values and also the related parameters for several averages used internally in the algorithm.

`SPO2.c` and `SPO2.h` contain the bulk of the algorithm code, which is described in the algorithm application note. `SPO2.h` defines many tunable parameters of the algorithm such as the duration of averaging used for the displayed respiration rate, as described in the algorithm application note.

`SAVGOL.c` and `SAVGOL.h` are files related to the high performance smoothing filter.

8. Compiling and Debugging Code

When importing an existing project, occasionally there are few things that need to be cleaned up.

First, clean the project removing existing compiled code. Then build the project.

1. **Project** > **Clean Targets**
2. **Project** > **Build Targets**

9. Loading code

Load code to the microcontroller using the method described in the EVK application note custom programming section.

9.1 Record Data

To record data from the UART output use a terminal program. For Windows SerialPlot is a good option, or TeraTerm. Typically two to five minutes of data is useful. At 100sps this means the data file is 12,000 to 30,000 lines long. Adjust the buffer in TeraTerm, if needed, to ensure all data is captured, or use SerialPlot or a Python script to capture serial prints.

A library of pre-recorded data for various user skin types, heart rates, breathing rates, perfusion index (finger temperature), etc., is invaluable in algorithm tuning and verification.

10. Algorithm Parameters and Tuning

Several settings and parameters should be tuned by customers based on the application, the optomechanical design, and the achieved signal-to-noise ratio. These parameters are found in spo2.h

10.1 SpO2 Calibration Parameters

10.1.1. R-curve

Spo2 calibration (R curve) is set by defined values `Rsquare`, `Rlinear`, and `Rconstant`, for example:

```
#define Rsquare (0)

#define Rlinear (-48)
```

```
#define Rconstant (118.5)
```

Spo2 is calculated according to the formula :

$$spo2 = Rsquare * R^2 + Rlinear * R + Rconstant$$

Where R is the ratio of ratios (Red AC/Red DC) / (IR AC/ IR DC).

Spo2 level is clipped below `MIN_SPO2` because such low values are not verifiable in a hypoxia lab. When `DEBUG_SPO2` is defined the clipping does not occur.

10.1.2. Finger Pressure Correction

For applications where variable finger pressure is of concern, customers can enable finger pressure correction by defining USE_PRESSURE_CORRECTION.

```
#define USE_PRESSURE_CORRECTION
```

To disable finger pressure correction, comment out this line.

Finger pressure correction is not the same for each individual due to variation in PPG waveform shapes, for instance, the size of the dicrotic notch and blood pressure. Therefore, finger pressure correction tuning parameters should be chosen conservatively. An example is provided below with explanatory comments.

```
#define MIN_PI_FOR_PRESSURE_CORRECTION 0.1 //no finger correction below this PI

#define PRESSURE_FACTOR_THRESHOLD 70 //pressure factor below this has no effect

#define PRESSURE_SLOPE_DENOM 9 //larges means less correction

#define PRESSURE_FACTOR_MAX 140 //pressure factor above this adds no correction
```

10.1.3. Noise Correction

To extend the useful range of PI, it is possible to subtract fixed noise from the data by defining `USE_NOISE_CORRECTION`. For details on characterizing noise, see the *OB1203 Algorithm Application Note*.

If the power supply noise dominates the SNR, the device will have noise that does not depend on LED intensity, but rather there is fixed noise associated with the receiver. For such applications, define the typical RMS (average deviation) noise with `TYP_NOISE`. This depends on the power settings (`BEST_SNR`, `MID_POWER`, or `LOW_POWER`) and the hardware. To reduce supply noise use an LDO on VDD, or in the case of the switching power supply use at least an RF choke resonant damper to reduce pulsation on VDD and VLED.

The RL2 and BT2 evaluation kits are power supply noise limited. *To achieve lower noise you can short the boost regulator PMIC input and output pins and power directly from batteries (provided the batteries are fresh) or a clean DC supply*.

If the application has noise that primarily scales with LED intensity (Tx noise limited) define `SCALE_TX_NOISE`. Then define the max noise at IR LED current setting of 1023.

```
#define MAX_NOISE 40 //Tx noise at max IR current (0x3F)
```

10.1.4. SG Filter Tuning

For hardware where the SNR is not ideal, if higher than expected SpO2 is observed, the SG filter length should be shortened as the additional noise affects the smoothed red channel amplitude more than the IR channel.

To avoid this effect, either change the calibration, or reduce the SG filter length. For the case of reduced SG filter length, in the following section of code, replace the short and long SG values by 1/2 of the nominal value. For example, replace 8 by 4. 4 indicates an actual SG filter length of 8.

```
#else
      #define SHORT_SG 4 //must be a power of 2
      #define LONG_SG 8 //must be a power of 2
      #define SHORT_PER1F_THRESH_DOWN 400 //120 BPM
      #define SHORT_PER1F_THRESH_UP 444 //110 BPM
#endif
```

In the same section of code, you can set the threshold for heart rate period (including 3 bits of fixed precision) for which we shorten the SG filter to allow higher bandwidth heart rate signals. For example, the threshold at which we switch to shorter SG filter occurs at 400, which is a heart rate period of 50 samples or 6000/50 = 120bpm.

11. Revision History

Revision	Date	Description
1.00	Feb 24, 2021	Initial release.