Getting started with BioVSM vital signs monitoring library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The BioVSM is a middleware library component of the X-CUBE-MEMS1 software and runs on STM32. It is able to monitor the followings: Band-pass filtered ECG voltage, Lead-off detection, N-peak detection, Most recent NN-interval value, Heartbeat rate (HR), Maximum heartbeat rate, Minimum heartbeat rate, Standard deviation of NN-intervals (SDNN), Root mean square of successive differences of NN-intervals (RMSSD), Standard deviation of successive differences of NN-intervals (SDSD), Percentage of successive differences of NN-intervals that are higher than 50 ms (PNN50) and Percentage of successive differences of NN-intervals that are higher than 20 ms (PNN20).

This library is intended to work with the ST1VAFE6AX and ST1VAFE3BX sensor only.

The algorithm is provided in static library format and is designed to be used on STM32 microcontrollers based on the ARM® Cortex® -M0+, ARM® Cortex®-M3, ARM® Cortex®-M33, ARM® Cortex®-M4 or ARM® Cortex®-M7 architectures.

It is built on top of STM32Cube software technology to ease portability across different STM32 microcontrollers.

1 Acronyms and abbreviations

Table 1. List of acronyms

Acronym	Description
API	Application programming interface
BSP	Board support package
GUI	Graphical user interface
HAL	Hardware abstraction layer
IDE	Integrated development environment

2 BioVSM middleware library in X-CUBE-MEMS1 software expansion for STM32Cube

2.1 BioVSM overview

The BioVSM library expands the functionality of the X-CUBE-MEMS1 software.

The library acquires data from the vAFE sensor and provides real-time information about Vital Signs Monitoring like ECG signal and parameters related to the heart rate and the heart rate variability of the user.

This library is intended to work with the ST1VAFE6AX and ST1VAFE3BX sensors only. Functionality and performance when using other sensors are not analyzed and can be significantly different from what described in the document.

2.2 BioVSM library

Technical information fully describing the functions and parameters of the BioVSM APIs can be found in the BioVSM_Package.chm compiled HTML file located in the Documentation folder.

2.2.1 BioVSM library description

The BioVSM sensor fusion library manages data acquired from the vafe sensor; it features:

- Band-pass filtered ECG voltage
- Lead-off detection
- N-peak detection
- Most recent NN-interval value
- Heartbeat rate (HR)
- Maximum heartbeat rate
- Minimum heartbeat rate
- Standard deviation of NN-intervals (SDNN)
- Root mean square of successive differences of NN-intervals (RMSSD)
- Standard deviation of successive differences of NN-intervals (SDSD)
- Percentage of successive differences of NN-intervals that are higher than 50 ms (PNN50)
- Percentage of successive differences of NN-intervals that are higher than 20 ms (PNN20)
- recommended sensor data sampling frequency of 240 Hz
- resources requirements:flash ram
- Cortex-M0+: 15.36 kB of code and up to 0.42 kB of data memory
- Cortex-M33: 14.97 kB of code and up to 0.39 kB of data memory
- Cortex-M3: 15.35 kB of code and up to 0.41 kB of data memory
- Cortex-M4: 14.96 kB of code and up to 0.41 kB of data memory
- Cortex-M7: 14.98 kB of code and up to 0.41 kB of data memory

Note: Size of dynamically allocated data memory is dependent on algorithm setup.

- available for ARM Cortex-M0+, Cortex-M3, Cortex-M33, Cortex-M4 and Cortex-M7 architecture

2.2.2 BioVSM library operation

The BioVSM library implements Vital Signs Monitoring and it features ECG signal computation, lead-off detection, N-peak detection, and computation of parameters related to heart rate and heart rate variability.

The library is designed for the ST1VAFE3BX and ST1VAFE6AX sensors only. Its functionality and performance with other sensors have not been analyzed and can differ significantly from what is described here.

The user should connect the two input pins of the vAFE sensor with an appropriate ECG acquisition system so that the electrical activity of the heart can be detected by the sensor. As an example:

- If the user is using the STEVAL-MKE006A electrode board with the ST1VAFE3BX's STEVAL-MKI250A adapter board of the STEVAL-MKI250KA kit or the STEVAL-MKE005A electrode board with the ST1VAFE6AX's STEVAL-MKI249A adapter board of the STEVAL-MKI249KA kit, the user may place a finger of each hand on one of the two electrodes
- If the user is using the STEVAL-MKE007A electrode board with the ST1VAFE3BX's STEVAL-MKI250A adapter board of the STEVAL-MKI250KA kit or the ST1VAFE6AX's STEVAL-MKI242A adapter board, the user may connect to the 3.5" jack the standard ECG electrodes, which should be placed on the standard chest / limb leads

The gain and the offset of the acquisition system should be known to the user from hardware characteristics and provided to the library for a correct operation of the library. The actual values of gain and offset also depend on the impedance matching due to the electrode-skin interface, but providing the nominal values may suffice. If they are completely unknown, lead off detection might need to be disabled, but in doing so the performance of the algorithms may be reduced. As an example:

- If the user is using the STEVAL-MKE006A electrode board with the ST1VAFE3BX's STEVAL-MKI250A adapter board, the nominal gain is 1 and the nominal offset is 0
- If the user is using the STEVAL-MKE005A electrode board with the ST1VAFE6AX's STEVAL-MKI249A adapter board, the nominal gain is 60 and the nominal offset is 0

Another parameter to be provided to the library is the sensitivity of the vAFE sensor in use. This value can be retrieved from the datasheet of each device. As an example:

- If the user is using the ST1VAFE3BX device and is setting the vAFE channel gain to 16, the sensitivity of the vAFE sensor is 1311 LSB/mV
- If the user is using the ST1VAFE6AX, the sensitivity of the vAFE sensor is 78 LSB/mV

The library implements algorithms to compute the ECG signal from the vAFE sensor input. Among these algorithms, a band-pass filter is applied. The cut-off frequencies of this band-pass filter can be changed with respect to their default values (0.5 Hz and 30 Hz). The execution rate of the library must be strictly greater than twice the high cut-off frequency of this filter (strictly greater than 60 Hz if the default value is used). A stricter constraint on the execution rate of the library may apply depending on the configured frequency of the notch filter, as detailed below.

If the acquisition system is connected to mains, a notch filter might optionally be enabled and configured to filter out the mains frequency from the ECG signal. If the acquisition system is never connected to mains, the notch filter should be disabled. If the notch filter is enabled and configured at 50 Hz, the execution rate of the library should be strictly greater than 102 Hz. If the notch filter is enabled and configured at 60 Hz, the execution rate of the library should be strictly greater than 122 Hz.

The library also performs a software implementation of the so-called "lead-off detection" feature to report if electrodes are not properly connected. When the ECG output exceeds the normal boundaries (+5 mV and +5 mV), the lead-off detection flag is set to true, the algorithms for peak detection and the computation of other parameters stops being executed and their output values are reset. When lead-off detection flag returns to false, the algorithms for peak detection and the computation of other parameters are re-initialized and starts being executed again. Lead-off detection can be disabled, but it is highly suggested to keep it enabled, as it is by default.

The library implements algorithms to perform peak detection to find the N-peaks (normal R-peaks) of the QRS-complex of the ECG signal, and it also outputs the value of the most recent NN-interval (time interval between to N-peaks). The library is robust with respect to missed peaks. If successive peaks are not properly detected for a sustained amount of time, the output values of the algorithms for peak detection and the computation of other parameters are reset. When a peak is detected, the peak detection flag is set to true, and all the outputs related to the computation of heart rate and heart rate variability parameters are updated.

Following peak detection, the library implements algorithms to compute heart rate parameters: heartbeat rate, maximum heartbeat rate and minimum heartbeat rate. The heartbeat rate is computed by applying a moving average to the values obtained at each NN-interval. The maximum and minimum heartbeat values are the highest and lowest values since the initialization / re-initialization of the algorithms.

The library also implements algorithms to compute some parameters that have been shown in scientific literature to correlate with heart rate variability. All these parameters are computed by applying a moving average to the values obtained at each NN-interval. The computed parameters are:

- SDNN (standard deviation of NN-intervals)
- RMSSD (root mean square of successive differences of NN-intervals)
- SDSD (standard deviation of successive differences of NN-intervals)
- PNN50 (percentage of successive differences of NN-intervals that are higher than 50 ms)
- PNN20 (percentage of successive differences of NN-intervals that are higher than 20 ms)

2.2.3 BioVSM library parameters

In the following, the types that are defined in the header file of the library.

```
typedef void * VSM_Instance_t;
```

- Pointer to the library instance loaded in data memory

```
typedef enum
{
  BIO_VSM_MCU_STM32 = 0,
  BIO_VSM_MCU_BLUE_NRG1,
  BIO_VSM_MCU_BLUE_NRG2,
  BIO_VSM_MCU_BLUE_NRG_LP,
  BIO_VSM_MCU_STM32WB0,
} VSM_mcu_type_t;
```

- Used MCU type

```
typedef enum
{
  BIO_VSM_INIT_OK = 0,            /* No error */
  BIO_VSM_INIT_ERR_INSTANCE,      /* Instance pointer NULL */
  BIO_VSM_INIT_ERR_ALREADY,       /* Instance already initialized */
  BIO_VSM_INIT_ERR_DEVICE_CONFIG, /* Invalid device configuration parameter */
  BIO_VSM_INIT_ERR_ALGO_CONFIG,   /* Invalid algorithm configuration parameter */
  BIO_VSM_INIT_ERR_RES,           /* Reserved error code */
} VSM_init_err_t;
```

- Library status - error code returned by `BioVSM_Start` API function

```
typedef enum
{
  BIO_VSM_DEINIT_OK = 0,       /* No error */
  BIO_VSM_DEINIT_ERR_INSTANCE, /* Instance pointer NULL */
  BIO_VSM_DEINIT_ERR_UNINIT,   /* Instance not initialised*/
  BIO_VSM_DEINIT_ERR_RES,      /* Reserved error code */
} VSM_deinit_err_t;
```

- Library status - error code returned by `BioVSM_deinit` API function

```
typedef enum
{
  BIO_VSM_RUN_OK = 0,       /* No error */
  BIO_VSM_RUN_ERR_INSTANCE, /* Instance NULL */
  BIO_VSM_RUN_ERR_UNINIT,   /* Instance not initialised*/
  BIO_VSM_RUN_ERR_RES,      /* Reserved error code */
} VSM_run_err_t;
```

- Library status - error code returned by `BioVSM_Update` API function

```
typedef enum
{
  BIO_VSM_MAINS_NONE = 0, /* No mains frequency (battery-powered device) */
  BIO_VSM_MAINS_50HZ = 1, /* 50 Hz mains frequency */
  BIO_VSM_MAINS_60HZ = 2, /* 60 Hz mains frequency */
} VSM_mains_t;
```

- Mains frequency

```
typedef struct
{
  uint16_t odr;
  uint16_t vafe_sens;
  float ecg_gain;
  float ecg_offset;
  VSM_mains_t mains;
} VSM_device_config_t;
```

- The parameters of the device, that must be configured and/or retrieved in the application code and passed to the algorithm during initialization:
  - `odr` – ODR in Hz, possible values are 240 Hz
  - `vafe_sens` – vAFE sensitivity [LSB/mV]
  - `ecg_gain` – ECG acquisition system gain [-], set to 1 if unknown
  - `ecg_offset` – ECG acquisition system offset [mV], set to 0 if unknown
  - `mains` – mains frequency

```
typedef struct
{
  float cutoff_low;
  float cutoff_high;
  bool lead_off_en;
  uint8_t peak_lat;
} VSM_algo_config_t;
```

- The parameters of the algorithm that can be configured from the application code:
  - `cutoff_low` – ECG band-pass filter low cut-off frequencies
  - `cutoff_high` – ECG band-pass filter high cut-off frequencies
  - `lead_off_en` – Lead-off detection enabling flag (performances are reduced when disabled)
  - `peak_lat` – Latency in showing N-peak detection output as number of consecutive valid peaks

```
typedef struct
{
  int16_t vafe;
} VSM_in_t;
```

- The inputs of the algorithms that must be provided to the algorithms at each iteration:
  - `vafe` – vAFE data [LSB]

```
typedef struct
{
  float ecg;
  bool lead_off;
  bool peak;
  float nn;
  uint8_t hbr;
  uint8_t hbr_max;
  uint8_t hbr_min;
  float sdnn;
  float rmssd;
  float sdsd;
  float pnn50;
  float pnn20;
} VSM_out_t;
```

- The outputs of the algorithms that are provided by the algorithms at each iteration:
  - `ecg` – Band-pass filtered ECG voltage [mV]
  - `lead_off` – Lead-off detection (if enabled, when this parameter is set to true, all the algorithms are reset)
  - `peak` – N-peak detection (in the samples where this is set to true, all the following parameters are updated)
  - `nn` – Most recent NN-interval value [ms]
  - `hbr` – Heartbeat rate [bpm]
  - `hbr_max` – Maximum heartbeat rate [bpm]
  - `hbr_min` – Minimum heartbeat rate [bpm]
  - `sdnn` – Standard deviation of NN-intervals [ms]
  - `rmssd` – Root mean square of successive differences of NN-intervals [ms]
  - `sdsd` – Standard deviation of successive differences of NN-intervals [ms]
  - `pnn50` – Percentage of successive differences of NN-intervals that are higher than 50 ms [%]
  - `pnn20` – Percentage of successive differences of NN-intervals that are higher than 20 ms [%]

2.2.4 BioVSM library parameters

In the following, the types that are defined in the header file of the library.

```
uint8_t BioVSM_GetLibVersion(char *version)
```

- Retrieve the version of the library
- `version` – pointer to an array of 35 characters
- Return the number of characters in the version string

```
void BioVSM_Initialize(VSM_mcu_type_t mcu_type)
```

- Perform BioVSM library initialization and setup of the internal mechanism

Note: This function must be called before using the vital signs monitoring library and the CRC module in STM32 microcontroller (in RCC peripheral clock enable register) has to be enabled.

- `mcu_type`

  – type of MCU used

```
VSM_Instance_t BioVSM_CreateInstance(VSM_algo_config_t *algo_conf)
```

- Create instance of BioVSM library algorithm:
  - Allocate the memory for a library instance
  - Fill the structure pointed by `algo_conf` with the default values of the parameters for the configuration of the algorithms
  - Return a pointer to its memory location

Note: After calling this function, the algorithms can be initialized with the `BioVSM_Start()` API function.

- `algo_conf` – configuration of the algorithm
- Return pointer to new instance of the algorithm

```
void BioVSM_DeleteInstance(VSM_Instance_t instance)
```

- Delete instance of BioVSM library algorithm:
  - De-initialize the algorithm
  - Free the memory allocated for the instance
- `instance` – pointer to instance of the algorithm to be deleted

```
VSM_init_err_t BioVSM_Start(VSM_Instance_t instance, VSM_device_config_t
                            *device_conf, VSM_algo_config_t *algo_conf
```

- Start the BioVSM engine:
  - Initialize (or re-initialize) the algorithm of the instance following the parameters set in the two structures pointed by `device_conf` and `algo_conf`
  - Return an initialization error code (e.g.: if invalid device parameters were set)
- `instance` – pointer to instance of the algorithm to be started
- `device_conf` – configuration of the device
- `algo_conf` – configuration of the algorithm
- Return an initialization error code

```
VSM_run_err_t BioVSM_Update(VSM_Instance_t instance, VSM_in_t *data_in, VSM_out_t *data_out);
```

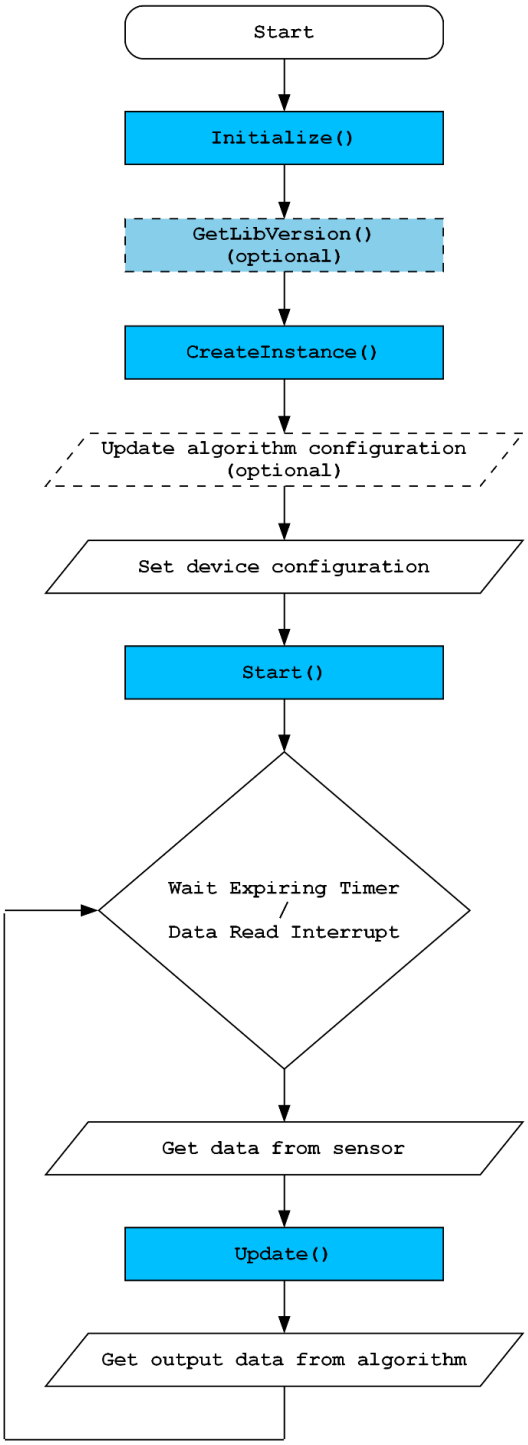
- Execute one step of the algorithm
- `instance` – pointer to instance of the algorithm
- `data_in` – input data
- `data_out` – output data

```
VSM_deinit_err_t BioVSM_deinit(VSM_Instance_t *instance)
```

- Deinitialize the algorithm of a library instance
- `instance` – pointer to instance of the algorithm to be deinitializted
- Return an deinitialization error code

2.2.5 API flow chart

Figure 1. **BioVSM API logic sequence**



2.2.6 Demo code

```
#define VSM_STR_LENG  35

[...]

/*** Initialization ***/

char lib_version[VSM_STR_LENG];
VSM_mcu_type_t mcu = BIO_VSM_MCU_STM32;
VSM_Instance_t VSM_Instance;
VSM_algo_config_t algo_conf;
VSM_device_config_t device_conf;
VSM_init_err_t init_status;
VSM_deinit_err_t deinit_status;
VSM_run_err_t run_err;

/* Library API initialization function */
BioVSM_Initialize(mcu);

/* Optional: Get version */
BioVSM_GetLibVersion(lib_version);

/* Create library algorithm instance */
VSM_Instance = BioVSM_CreateInstance(&algo_conf);

/* Detail the hardware configuration (in this example, ST1VAFE6AX device with STEVAL-MKE005A
electrode board) */
device_conf.odr = 240;
device_conf.vafe_sens = 78;
device_conf.ecg_gain = 60.0f;
device_conf.ecg_offset = 0.0f;
device_conf.mains = VSM_MAINS_50HZ;

/* Start the algorithm engine */
init_status = BioVSM_Start(instance, &device_conf, &algo_conf);

/*** Using Vital Signs Monitoring algorithm ***/
Timer_OR_DataRate_Interrupt_Handler()
{
  VSM_in_t data_in;
  VSM_out_t data_out;

  /* Get data from sensor */
  ReadSensor(&data_in.vafe);

  /* Execute one step of the algorithms */
  run_err = BioVSM_Update(instance, &data_in, &data_out)

  /* Get output data from algorithm */
  uint8_t hbr = data_out.hbr;
  float ecg = data_out.ecg;
  [...]
}

/* End the algorithm engine */
deinit_status = BioVSM_deinit(instance)
```

2.2.7 Algorithm performance

Table 2. Elapsed time (µs) algorithm

Min	Min.	Average	Max.
Cortex-M4 STM32F401RE at 84 MHz	28	28.3	31

2.3 References

All of the following resources are freely available on www.st.com.

[1] Getting started with the X-CUBE-MEMS1 motion MEMS and environmental sensor software expansion for STM32Cube (UM1859)

[2] STM32 Nucleo-64 board (UM1724)

[3] Getting started with Unicleo-GUI for motion MEMS and environmental sensor software expansion for STM32Cube (UM2128)

Revision history

Table 3. **Document revision history**

Date	Revision	Changes
04-Feb-2025	1	Initial release.

Contents

Disclaimer

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.