
Getting started with MotionVC vertical context library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionVC is a middleware library part of [X-CUBE-MEMS1](#) software and runs on STM32. It provides real-time information about vertical movement. The library is able to detect a change of altitude and distinguish the type of vertical movement: stairs, elevator, and escalator.

This library is intended to work with ST MEMS only.

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

It is built on top of [STM32Cube](#) software technology that eases portability across different STM32 microcontrollers.

The software comes with sample implementation running on [X-NUCLEO-IKS01A3](#) and [X-NUCLEO-IKS4A1](#) expansion board on a [NUCLEO-F401RE](#), [NUCLEO-L152RE](#) or [NUCLEO-U575ZI-Q](#) development board.

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 MotionVC middleware library for X-CUBE-MEMS1 software expansion for STM32Cube

2.1 MotionVC overview

The MotionVC library expands the functionality of the [X-CUBE-MEMS1](#) software.

The library acquires data from the accelerometer and pressure sensor, detects changes of altitude and distinguishes type of vertical movement: stairs, elevator, and escalator. The library is able to adapt to different noise floor of the pressure sensor data.

The library is designed for ST MEMS only. Functionality and performance when using other MEMS sensors are not analyzed and can be significantly different from what described in the document.

Sample implementation is available for the [X-NUCLEO-IKS01A3](#) and [X-NUCLEO-IKS4A1](#) expansion board, mounted on a [NUCLEO-F401RE](#), [NUCLEO-L152RE](#) or [NUCLEO-U575ZI-Q](#) development board.

2.2 MotionVC library

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

2.2.1 MotionVC library description

The MotionVC vertical context library manages data acquired from accelerometer and pressure sensor; it features:

- vertical movement detection: on floor, up/down
- type of vertical movement detection: stairs, elevator, escalator
- drift free altitude and vertical velocity with confidence parameter calculation
- inbuilt step detection mode
- automatic adaptation to different noise floor of pressure sensor data
- required accelerometer data sampling frequency of 50 Hz and pressure sensor sampling frequency of 10 Hz
- resources requirements:
 - Cortex-M4: 12.2 kB of code and 4.0 kB of data memory
 - Cortex-M3: 12.5 kB of code and 4.0 kB of data memory
 - Cortex-M33: 12.1 kB of code and 4.0 kB of data memory
 - Cortex-M7: 12.3 kB of code and 4.0 kB of data memory
- available for ARM Cortex-M4, M3, M33 and M7 architectures

2.2.2 MotionVC APIs

The MotionVC library APIs are:

- `uint8_t MotionVC_GetLibVersion(char *version)`
 - retrieves the version of the library
 - `*version` is a pointer to an array of 35 characters
 - returns the number of characters in the version string
- `void MotionVC_Initialize(void)`
 - performs MotionVC library initialization and setup of the internal mechanism

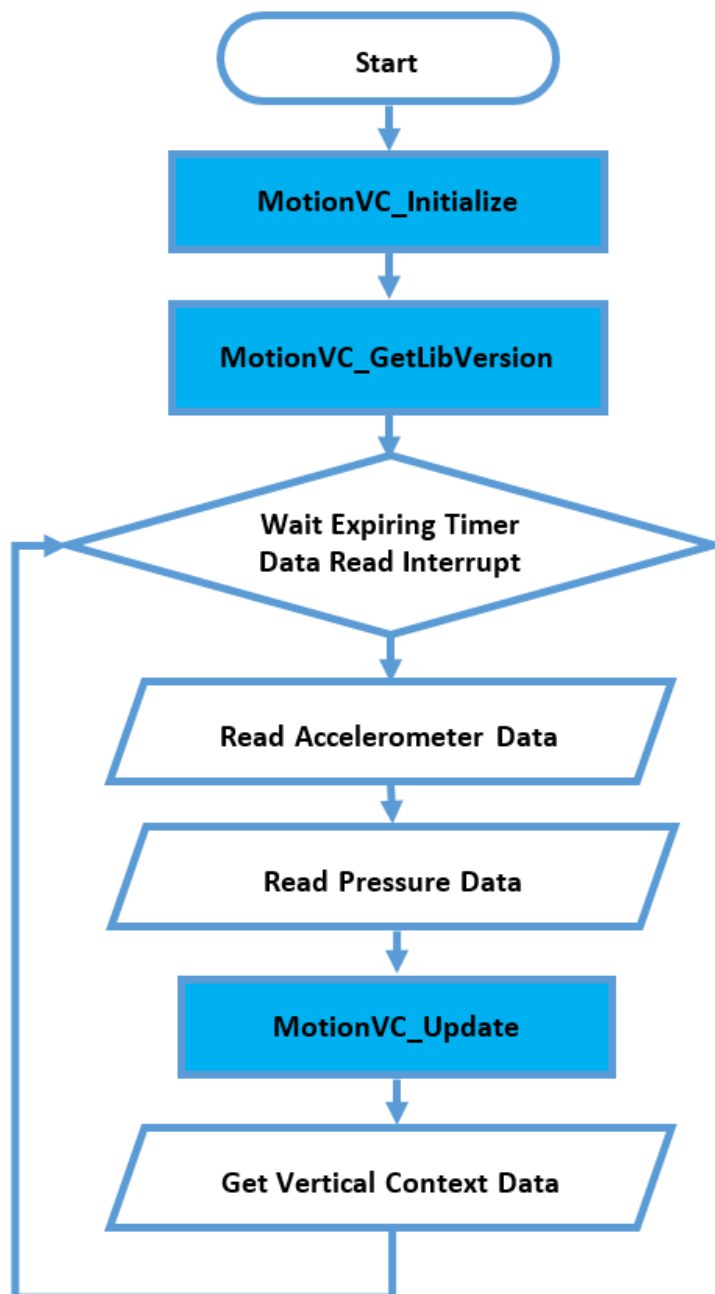
Note: This function must be called before using the vertical context library

- `void MotionVC_Update(MVC_input_t *data_in, MVC_output_t *data_out)`
 - runs the vertical context and altitude determination algorithm
 - retrieves the latest vertical context results
 - the parameters for the structure type `MVC_input_t` are:
 - `AccX` is the acceleration in X axis in g
 - `AccY` is the acceleration in Y axis in g
 - `AccZ` is the acceleration in Z axis in g
 - `Press` is the pressure sensor data in hPa
 - the parameters for the structure type `MVC_output_t` are:
 - `Timestamp` is the timestamp
 - `Valid` is the flag that indicates if the result is valid or not
 - `Baro_Altitude` is the altitude in cm computed from pressure using the standard formula
 - `Cal_Altitude` is the calibrated altitude in cm with the drift correction
 - `Speed` is the structure (`MVC_speed_t`) with information about vertical speed
 - `Context` is the vertical context (`MVC_context_t`)
 - `Confidence` is the confidence in the context (`MVC_confidence_t`)
 - `NSteps` is the number of detected steps
 - the parameters for the structure type `MVC_speed_t` are:
 - `Speed` is the vertical speed in cm/s
 - `Speed_Error` is the estimated error of the vertical speed in cm/s
 - the items for the enum type `MVC_context_t` are:
 - `MVC_UNKNOWN` value for no pressure data or reliable data
 - `MVC_FLOOR` value for walking on flat surface
 - `MVC_UPDOWN` value for significant change observed in height
 - `MVC_STAIRS` value for stairs
 - `MVC_ELEVATOR` value for elevator
 - `MVC_ESCALATOR` value for escalator
 - the items for the enum type `MVC_confidence_t` are:
 - `MVC_CONFIDENCE_UNKNOWN`
 - `MVC_CONFIDENCE_POOR`
 - `MVC_CONFIDENCE_MED`
 - `MVC_CONFIDENCE_HIGH`

Note: *This function has to be called periodically.*

2.2.3 API flow chart

Figure 1. MotionVC API logic sequence



2.2.4 Demo code

The following demonstration code reads data from the accelerometer and pressure sensor and calculates vertical context data.

```
[...]

#define    VERSION_STR LENG    35

/** Initialization **/

char lib_version[VERSION_STR LENG];

/* Vertical Context API initialization function */ MotionVC_Initialize();
/* Optional: Get version */
MotionVC_GetLibVersion(lib_version);

[...]

/** Using vertical context algorithm **/

Timer_OR_DataRate_Interrupt_Handler()

{

    MVC_input_t data_in;
    MVC_output_t data_out;

    /* Get acceleration X/Y/Z in g */
    MEMS_Read_AccValue(&data_in.AccX, &data_in.AccY, &data_in.AccZ);

    /* Get pressure in hPa */
    MEMS_Read_PressValue(&data_in.Press);

    /* Vertical context algorithm update */
    MotionVC_Update(&data_in, &data_out);

}
```

2.2.5 Algorithm performance

Table 2. Cortex-M4 and Cortex-M3: elapsed time (μs) algorithm

Cortex-M4 STM32F401RE at 84 MHz			Cortex-M3 STM32L152RE at 32 MHz		
Min	Avg	Max	Min	Avg	Max
2	150	1125	5	530	4585

Table 3. Cortex-M33 and Cortex-M7: elapsed time (μs) algorithm

Cortex-M33 STM32U575ZI-Q at 160 MHz			Cortex-M7 STM32F767ZI at 96 MHz		
Min	Avg	Max	Min	Avg	Max
1	120	814	5	581	3457

2.3 Sample application

The MotionVC middleware can be easily manipulated to build user applications; a sample application is provided in the Application folder.

It is designed to run on a [NUCLEO-F401RE](#), [NUCLEO-L152RE](#) or [NUCLEO-U575ZI-Q](#) development board connected to an [X-NUCLEO-IKS01A3](#) and [X-NUCLEO-IKS4A1](#) expansion board.

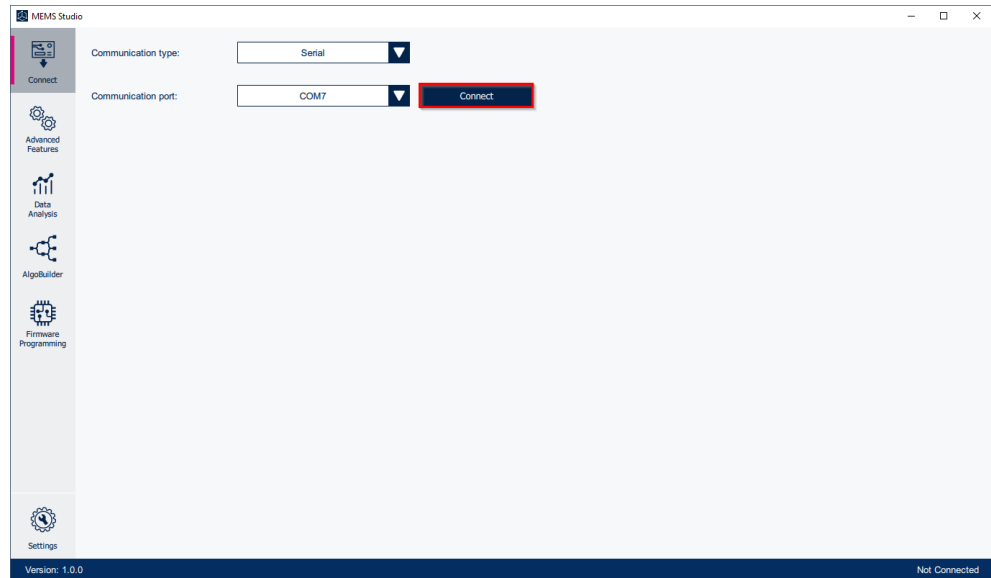
The application detects vertical context in real-time. Data can be displayed through a GUI. USB cable connection is required to monitor real-time data and to power the board from the PC via USB connection.

2.3.1 MEMS Studio

The sample application uses the [MEMS-Studio](#) GUI application, which can be downloaded from www.st.com.

- Step 1.** Ensure that the necessary drivers are installed and the STM32 Nucleo board with appropriate expansion board connected to the PC.
- Step 2.** Launch the MEMS-Studio application to open the main application window.
If an STM32 Nucleo board with supported firmware is connected to the PC, it is automatically detected the appropriate COM port. Press **Connect** button to open this port.

Figure 2. MEMS-Studio - Connect



- Step 3.** When connected to STM32 Nucleo board with supported firmware *Library Evaluation* tab is opened.



To start and stop data streaming toggle the appropriate  start /  stop button on the outer vertical tool bar.
The data coming from the connected sensor can be viewed selecting the *Data Table* tab on the inner vertical tool bar.

Figure 3. MEMS-Studio - Library Evaluation - Data Table



- Step 4.** Select the *Vertical Context* tab on the inner vertical tool bar to open the dedicated application view. On the right you can see a visual representation of the current state and on the left there is a table containing information about all detected changes.

Figure 4. MEMS-Studio - Library Evaluation – Vertical Context



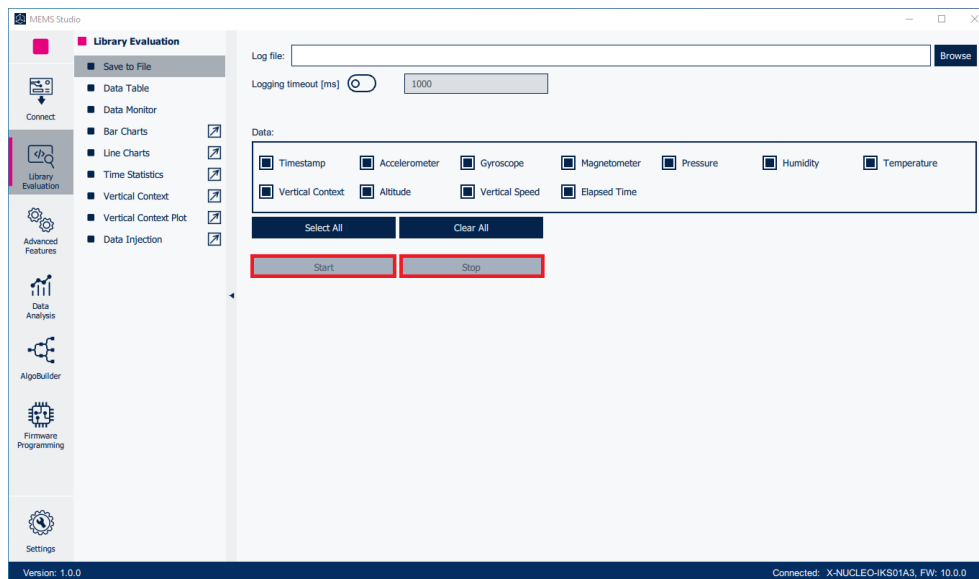
- Step 5.** Click on the *Vertical Context Plot* tab in the inner vertical toolbar to open dedicated view where you can see all the outputs from the vertical context algorithm. The window is split into three sections: the first one is the graph of standard and calibrated altitude, the second one is a graph of vertical speed and the third contains all the outputs from the algorithm (vertical context, altitude, vertical speed, number of steps, context confidence, calibrated altitude, vertical speed error, data valid flag) in text form.

Figure 5. MEMS-Studio - Library Evaluation – Vertical Context Plot



- Step 6.** Select the *Save to File* tab on the inner vertical tool bar to open the data logging configuration window. Select which sensor and activity data to save to the log file. You can start or stop saving by clicking the corresponding Start / Stop button.

Figure 6. MEMS-Studio - Library Evaluation - Save to File



2.4

References

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

1. [UM1859](#): Getting started with the X-CUBE-MEMS1 motion MEMS and environmental sensor software expansion for STM32Cube
2. [UM1724](#): STM32 Nucleo-64 boards (MB1136)
3. [UM3233](#): Getting started with MEMS-Studio

Revision history

Table 4. Document revision history

Date	Version	Changes
17-Sep-2018	1	Initial release.
15-Feb-2019	2	Updated Table 2. Elapsed time (μ s) algorithm. Added X-NUCLEO-IKS01A3 expansion board compatibility informaton.
26-Mar-2020	3	Updated Introduction, Section 2.2.1: MotionVC library description and Section 2.2.5: Algorithm performance .
18-Mar-2024	4	Updated Section Introduction , Section 2.1: MotionVC overview , Section 2.2: MotionVC library , Section 2.3: Sample application and Section 2.3.1: MEMS Studio .

Contents

1	Acronyms and abbreviations	2
2	MotionVC middleware library for X-CUBE-MEMS1 software expansion for STM32Cube	3
2.1	MotionVC overview	3
2.2	MotionVC library	3
2.2.1	MotionVC library description	3
2.2.2	MotionVC APIs	3
2.2.3	API flow chart	5
2.2.4	Demo code	5
2.2.5	Algorithm performance	6
2.3	Sample application	6
2.3.1	MEMS Studio	6
2.4	References	9
	Revision history	10

List of tables

Table 1.	List of acronyms	2
Table 2.	Cortex-M4 and Cortex-M3: elapsed time (μ s) algorithm.	6
Table 3.	Cortex-M33 and Cortex-M7: elapsed time (μ s) algorithm.	6
Table 4.	Document revision history	10

List of figures

Figure 1.	MotionVC API logic sequence	5
Figure 2.	MEMS-Studio - Connect	7
Figure 3.	MEMS-Studio - Library Evaluation - Data Table.	7
Figure 4.	MEMS-Studio - Library Evaluation – Vertical Context	8
Figure 5.	MEMS-Studio - Library Evaluation – Vertical Context Plot.	8
Figure 6.	MEMS-Studio - Library Evaluation - Save to File	9

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.

© 2024 STMicroelectronics – All rights reserved