UM2225

User manual

Getting started with MotionEC real-time E-Compass library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionEC is a middleware library component of the X-CUBE-MEMS1 software and runs on STM32. It provides real-time information about the device orientation and movement status based on data from a device.

It provides the following outputs: device orientation (quaternions, Euler angles), device rotation (virtual gyroscope functionality), gravity vector and linear acceleration.

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 and ARM® Cortex®-M4 architectures.

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

The software comes with sample implementation running on X-NUCLEO-IKS01A1 (with optional STEVAL-MKI160V1) or X-NUCLEO-IKS01A2 expansion board on a NUCLEO-F401RE, NUCLEO-L476RG or NUCLEO-L152RE 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 MotionEC middleware library in X-CUBE-MEMS1 software expansion for STM32Cube

2.1 MotionEC overview

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

The library acquires data from the accelerometer and magnetometer and provides information about the device orientation and movement status based on data from a device.

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.

A sample implementation is available on X-NUCLEO-IKS01A2 and X-NUCLEO-IKS01A1 (with optional STEVAL-MKI160V1) expansion boards, mounted on a NUCLEO-F401RE, NUCLEO-L476RG or NUCLEO-L152RE development board.

2.2 MotionEC library

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

2.2.1 MotionEC library description

The MotionEC E-Compass library manages data acquired from the accelerometer and magnetometer; it features:

- device orientation (quaternions, Euler angles), device rotation (virtual gyroscope functionality), gravity vector and linear acceleration outputs
- functionality based on the accelerometer and magnetometer data only
- required accelerometer and magnetometer data sampling frequency of up to 100 Hz
- 3 kByte of code and 0.1 kByte of data memory usage

  Note: Real size might differ for different IDEs (toolchain)
- available for ARM Cortex-M3 and Cortex-M4 architectures

2.2.2 MotionEC APIs

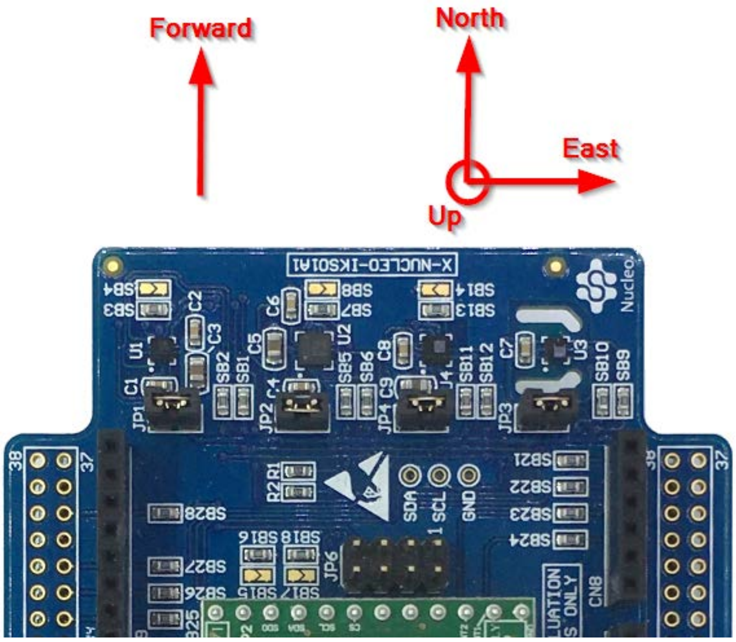
The MotionEC APIs are:

- `uint8_t MotionEC_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 MotionEC_Initialize(float freq)`
  - performs MotionEC library initialization and setup of the internal mechanism.
  - the CRC module in STM32 microcontroller (in RCC peripheral clock enable register) has to be enabled before using the library
  - `freq` is the sensor sampling frequency [Hz]

  Note: This function must be called before using the E-Compass library.
- `void MotionEC_SetFrequency(float freq)`
  - sets the sampling frequency (modifying the filtering parameters)
  - `freq` is the sensor sampling frequency [Hz]
- `void MotionEC_Run(MEC_input_t *data_in, MEC_output_t *data_out)`
  - runs the E-Compass algorithm (accelerometer and magnetometer data fusion)
  - `*data_in` is a pointer to a structure with input data

- the parameters for the structure type `MEC_input_t` are:
  - `Acc[3]` is an array of accelerometer data in ENU convention, measured in g
  - `Mag[3]` is an array of magnetometer calibrated data in ENU convention, measured in μT/50
  - `DTime` is the delta time (i.e., time delay between old and new data set) measured in s
- `*data_out` is a pointer to a structure with output data
- the parameters for the structure type `MEC_output_t` are:
  - `Quaternion[4]` is array containing quaternion in ENU convention, representing the 3D-angular orientation of the device in the space; order of elements is: X, Y, Z, W, with always positive element W
  - `Euler[3]` is an array of Euler angles in ENU convention, representing the 3D-angular orientation of the device in space; the order of the elements is: yaw, pitch, roll, measured in deg
  - `IGyro[3]` is an array of angular rates in ENU convention, representing a virtual gyroscope sensor, measured in dps
  - `Gravity[3]` is an array of accelerations in ENU convention, representing the gravity vector, measured in g
  - `Linear[3]` is an array of accelerations in ENU convention, representing the device linear acceleration, measured in g

Figure 1. ENU reference frame

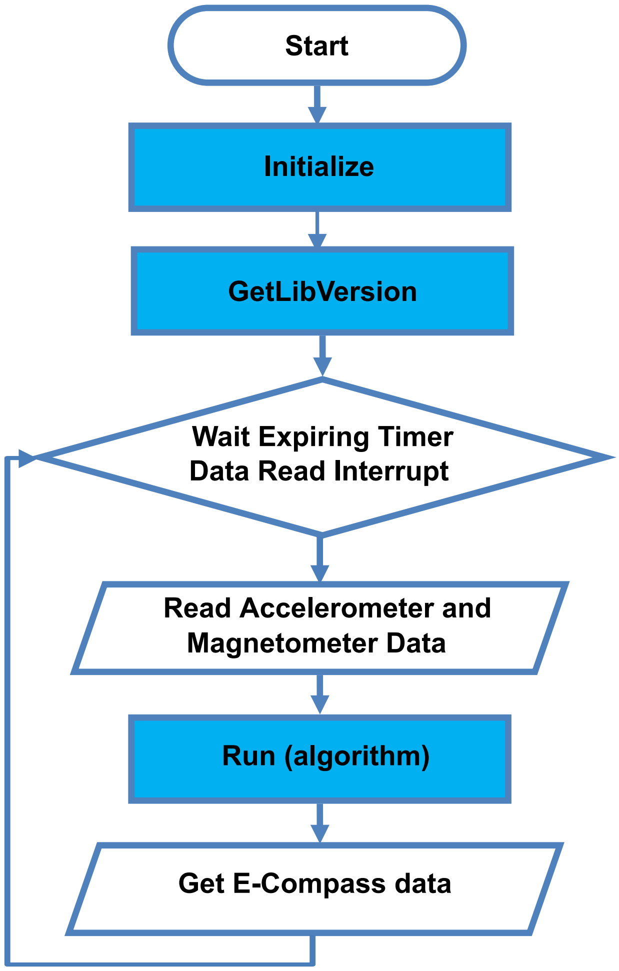


- `void MotionEC_GetOrientationEnable(MEC_state_t *state)`
  - gets the enable/disable state of the Euler angle calculation
  - `*state` is a pointer to the current enable/disable state
- `void MotionEC_SetOrientationEnable(MEC_state_t state)`
  - sets the enable/disable state of the Euler angle calculation
  - `state` is the new enable/disable state to be set
- `void MotionEC_GetVirtualGyroEnable(MEC_state_t *state)`
  - gets the enable/disable state of the virtual gyroscope calculation
  - `*state` is a pointer to the current enable/disable state
- `void MotionEC_SetVirtualGyroEnable(MEC_state_t state)`

  - sets the enable/disable state of the virtual gyroscope calculation
  - `state` is the new enable/disable state to be set
- `void MotionEC_GetGravityEnable(MEC_state_t *state)`
  - gets the enable/disable state of the gravity vector calculation
  - `*state` is a pointer to the current enable/disable state
- `void MotionEC_SetGravityEnable(MEC_state_t state)`
  - sets the enable/disable state of the gravity vector calculation
  - `state` is the new enable/disable state to be set
- `void MotionEC_GetLinearAccEnable(MEC_state_t *state)`
  - gets the enable/disable state of the linear acceleration calculation
  - `*state` is a pointer to the current enable/disable state
- `void MotionEC_SetLinearAccEnable(MEC_state_t state)`
  - sets the enable/disable state of the linear acceleration calculation
  - `state` is the new enable/disable state to be set

2.2.3 API flow chart

Figure 2. MotionEC API logic sequence



2.2.4 Demo code

The following demonstration code reads data from the accelerometer and magnetometer sensors and gets the E-Compass data (i.e., quaternion, Euler angles, etc.).

```
[…]
#define VERSION_STR_LENG 35
#define SAMPLE_FREQ      50
[…]

/*** Initialization ***/
char lib_version[VERSION_STR_LENG];

/* E-Compass API initialization function */
MotionEC_Initialize(SAMPLE_FREQ);

/* Optional: Get version */
MotionEC_GetLibVersion(lib_version);
```

```
[…]

/*** Using E-Compass algorithm ***/
Timer_OR_DataRate_Interrupt_Handler()
{
MEC_input_t  data_in;
MEC_output_t data_out;

/* Get acceleration X/Y/Z in [g] */
MEMS_Read_AccValue(&data_in.Acc[0], &data_in.Acc[1], &data_in.Acc[2]);

/* Get calibrated magnetic intensity X/Y/Z in [µT/50] */
MEMS_Read_CalibratedMagValue(&data_in.Mag[0], &data_in.Mag[1], &data_in.Mag[2]);

/* Set delta time in [s] */
data_in.Dtime = 1.0f / SAMPLE_FREQ;

/* Run E-Compass algorithm */
MotionEC_Run(&data_in, &data_out);
}
```

2.2.5 Algorithm performance

The E-Compass algorithm uses data from the accelerometer and magnetometer only. It runs at a low frequency (up to 100 Hz) to reduce power consumption.

Table 2. Elapsed time (µs) algorithm

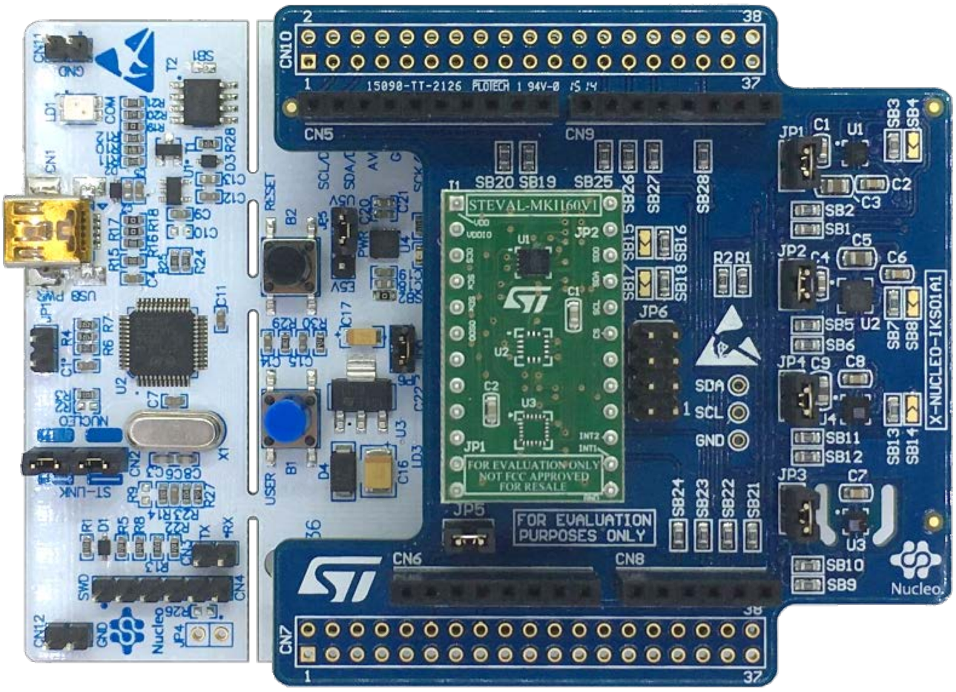
Cortex-M4 STM32F401RE at 84 MHz									Cortex-M3 STM32L152RE at 32 MHz								
SW4STM32 1.13.1 (GCC 5.4.1)			IAR EWARM 7.80.4			Keil µVision 5.22			SW4STM32 1.13.1 (GCC 5.4.1)			IAR EWARM 7.80.4			Keil µVision 5.22		
Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
249	264	277	175	176	273	175	471	475	554	1297	1349	860	865	1305	855	2024	2039

3 Sample application

The MotionEC 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-L476RG or NUCLEO-L152RE development board connected to an X-NUCLEO-IKS01A1 (based on LSM6DS0) or an X-NUCLEO-IKS01A2 (based on LSM6DSL) expansion board, with optional STEVAL-MKI160V1 board (based on LSM6DS3).

Figure 3. Sensor expansion board and adapter connected to the STM32 Nucleo



The application recognizes the device orientation and rotation in real-time. The data can be displayed through a GUI.

The algorithm provides the following outputs: device orientation (quaternions, Euler angles), device rotation (virtual gyroscope functionality), gravity vector and linear acceleration.

3.1 Unicleo-GUI application

The sample application uses the Windows Unicleo-GUI utility, 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 is connected to the PC.

Step 2. Launch the Unicleo-GUI application to open the main application window.

If an STM32 Nucleo board with supported firmware is connected to the PC, it is automatically detected and the appropriate COM port is opened.

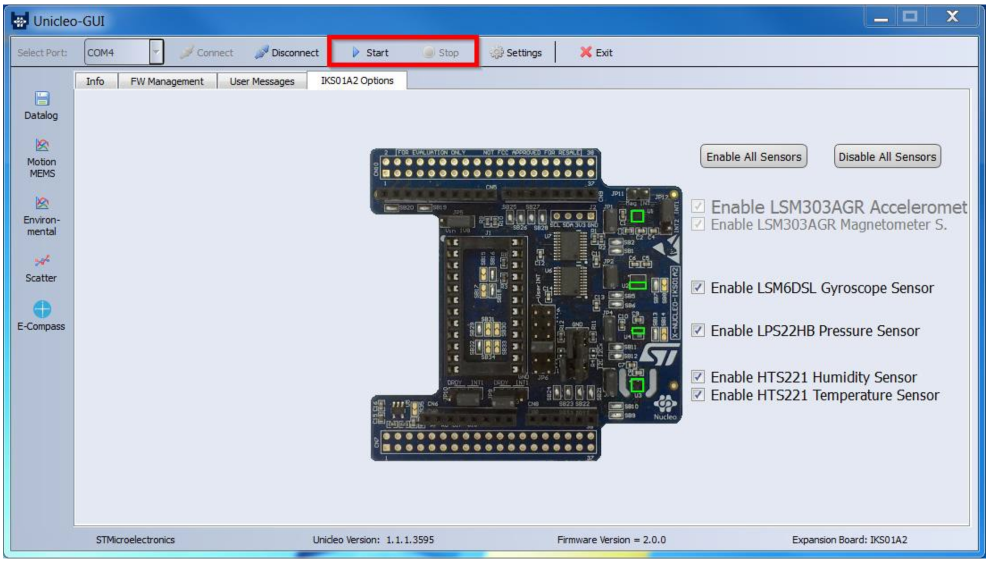


Figure 4. Unicleo main window

Step 3. Start and stop data streaming by using the appropriate buttons on the vertical tool bar. The data coming from the connected sensor can be viewed in the User Messages tab.

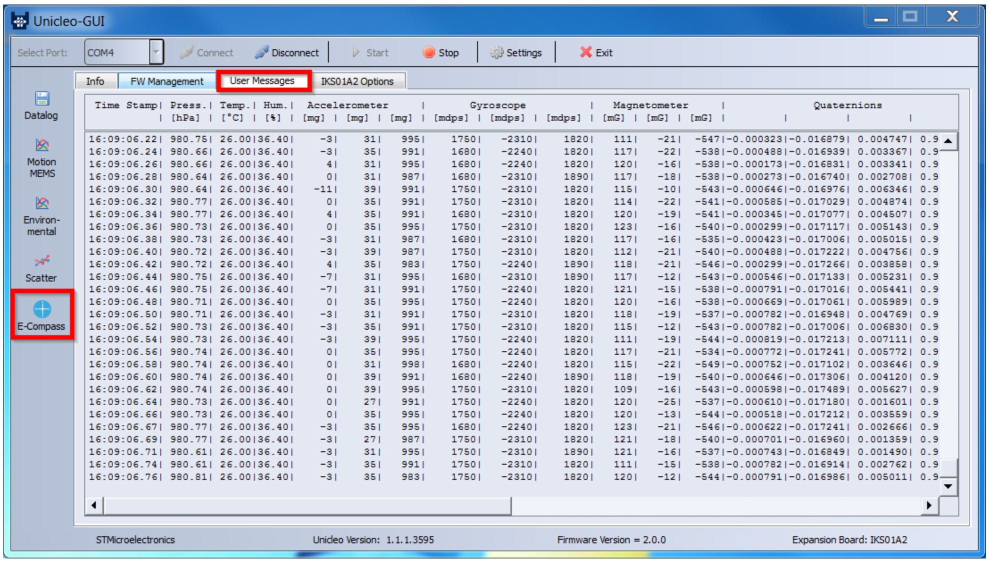
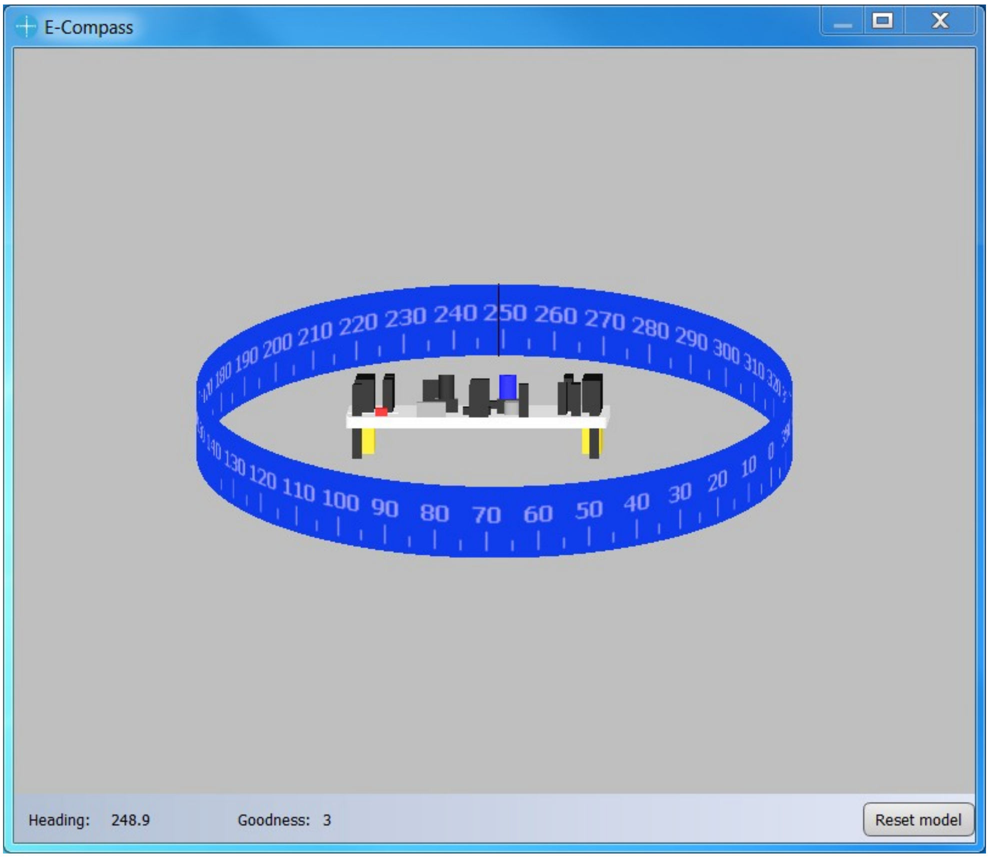


Figure 5. User Messages tab

Step 4. Click on the **E-Compass** icon in the vertical toolbar to open the dedicated application window.

Figure 6. **E-Compass window**



The figure above shows an STM32 Nucleo graphical model. The model orientation and rotation are based on E-Compass data (quaternions) calculated by the algorithm.

To align the real device movement with the graphical model, point the device (in the "forward direction" shown in) towards the screen and push the **Reset model**.

The heading value represents the real device heading.

Pointing the device straight up or down (along *Up* axis of ENU reference frame, with ±5 degree tolerance) gives *N/A* value for the heading: it is not possible to distinguish to which cardinal point the device is pointing to.

The goodness value gives 0 to 3 values and is related to the magnetometer calibration: the higher the value, the better the results of the E-Compass data algorithm.

Step 5. Click on the **Datalog** icon in the vertical toolbar to open the datalog configuration window: you can select the sensor and fusion data to be saved in the files. You can start or stop saving by clicking on the corresponding button.

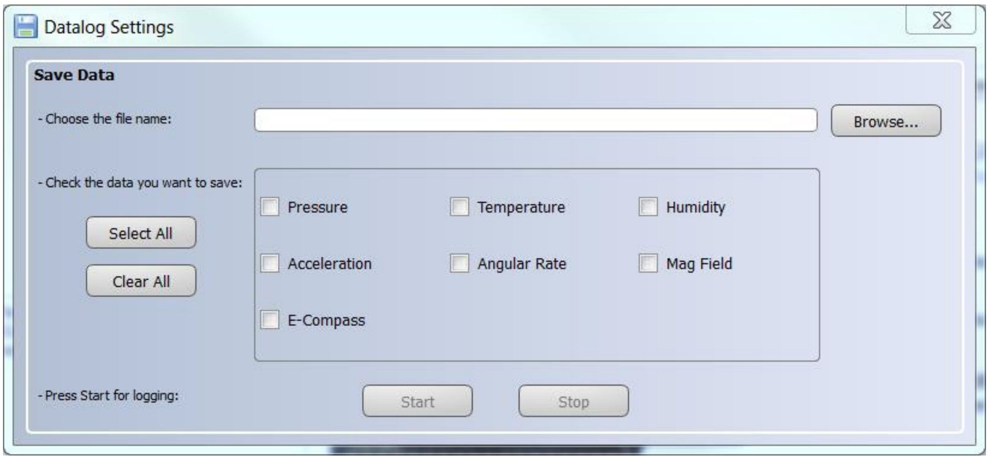


Figure 7. Datalog window

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 board
3. UM2128: Getting started with Unicleo-GUI for motion MEMS and environmental sensor software expansion for STM32Cube

Revision history

Table 3. **Document revision history**

Date	Version	Changes
18-May-2017	1	Initial release.
25-Jan-2018	2	Added refences to NUCLEO-L152RE development board and Table 2. Elapsed time (µs) algorithm.
21-Mar-2018	3	Updated Section • Introduction and Section 2.1 MotionEC overview.

Contents

List of tables

List of figures