
Getting started with MotionMC magnetometer calibration library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionMC is a middleware library part of [X-CUBE-MEMS1](#) software and runs on STM32. It provides real-time magnetometer calibration using hard iron (HI) and scale factor coefficients to correct magnetometer data.

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[®]-M0+, Cortex[®]-M3, 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 an [X-NUCLEO-IKS01A2](#), [X-NUCLEO-IKS01A3](#) or [X-NUCLEO-IKS02A1](#) expansion board on a [NUCLEO-F401RE](#), [NUCLEO-L476RG](#), [NUCLEO-L152RE](#) or [NUCLEO-L073RZ](#) 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 MotionMC middleware library in X-CUBE-MEMS1 software expansion for STM32Cube

2.1 MotionMC overview

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

Near the surroundings of a magnetometer, it can cause distortion of Earth magnetic field. Based on the distortion character, we can categorize it as two types: hard iron and soft iron effect.

Hard iron distortion arises from permanent magnets, magnetized iron or steel located nearby the magnetometer. This distortion remains constant and in fixed location related to the magnetometer for all heading orientations. Hard iron effect adds a constant magnitude field component along each magnetometer axis.

Soft iron distortion arises from the interaction of the Earth magnetic field and the material which surrounds the magnetometer sensor distorting the Earth magnetic field. The distortion magnitude and direction depend on the incident angle of the Earth magnetic field on the material. Hence, it varies with the magnetometer orientation.

The library acquires data from the magnetometer and calculates the hard iron (HI) and soft iron (SI) coefficients or, in the case of the library for Cortex-M3, M4 and M7 only the scale factor (SF - diagonal elements of soft iron matrix) coefficients together with the calibration quality value. The calibration coefficients are then used to compensate raw data from the magnetometer and reduce hard iron and soft iron effects.

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 on [X-NUCLEO-IKS01A2](#), [X-NUCLEO-IKS01A3](#) and [X-NUCLEO-IKS02A1](#) expansion boards, mounted on a [NUCLEO-F401RE](#), [NUCLEO-L476RG](#), [NUCLEO-L152RE](#) or [NUCLEO-L073RZ](#) development board.

2.2 MotionMC library

Technical information fully describing the functions and parameters of the MotionMC APIs can be found in the [MotionMC_Package.chm](#) compiled HTML file located in the Documentation folder.

2.2.1 MotionMC library description

The MotionMC magnetometer calibration library manages data acquired from a magnetometer; it features:

- update frequency up to 100 Hz
- hard iron compensation, theoretically unlimited (within sensor range)
- wide scale factor compensation range from 0.65 to 1.35 in every direction
- support for all ST magnetometer devices
- resources requirements:
 - Cortex-M0+: 10.1 kB of code and 1.7 kB of data memory
 - Cortex-M3: 24.7 kB of code and 1.4 kB of data memory
 - Cortex-M4: 22.4 kB of code and 1.4 kB of data memory
 - Cortex-M7: 21.8 kB of code and 1.4 kB of data memory
- available for ARM Cortex-M0+, Cortex-M3, Cortex-M4 and Cortex-M7 architectures

2.2.2 MotionMC APIs

The MotionMC APIs are:

- **Cortex-M3 and Cortex-M4**

- `uint8_t MotionMC_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 MotionMC_Initialize(int sampletime, unsigned short int enable)`
 - performs MotionMC library initialization and setup of the internal mechanism.

Note: This function must be called before using the magnetometer calibration library.

- `sampletime` parameter is the interval between update function calls in ms.
- `enable` parameter enables (1) or disables (0) the library
- the CRC module in STM32 microcontroller (in RCC peripheral clock enable register) has to be enabled before using the library
- `void MotionMC_Update (MMC_Input_t *data_in)`
 - runs the magnetometer calibration algorithm
 - `*data_in` parameter is a pointer to a structure with input data
 - the parameters for the structure type `MMC_Input_t` are:
 - `Mag[3]` is an array of magnetometer sensor value in μT
 - `TimeStamp` parameter is the timestamp value in ms for current magnetometer sensor data

Note: This function has to be called periodically at the same period that is indicated in the initialization function.

- `void MotionMC_GetCalParams (MMC_Output_t *data_out)`
 - retrieves the magnetometer hard iron (HI) and scale factor (SF) coefficients
 - `*data_out` parameter is a pointer to a structure with output data
 - the parameters for the structure type `MMC_Output_t` are:
 - `HI_Bias[3]` is an array of magnetometer hard iron (HI) coefficients in μT
 - `SF_Matrix[3][3]` is a 3x3 matrix of magnetometer scale factor (SF) coefficients
 - `CalQuality` is the calibration quality factor.

- **Cortex-M0+:**

- `uint8_t MotionMC_CM0P_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 *MotionMC_CM0P_Initialize(MMC_CM0P_mcu_type_t mcu_type void *mmc_algo, MMC_CM0P_Mode_t mode, unsigned short int enable)`
 - performs MotionMC library initialization and setup of the internal mechanism.

Note: *This function must be called before using the magnetometer calibration library.*

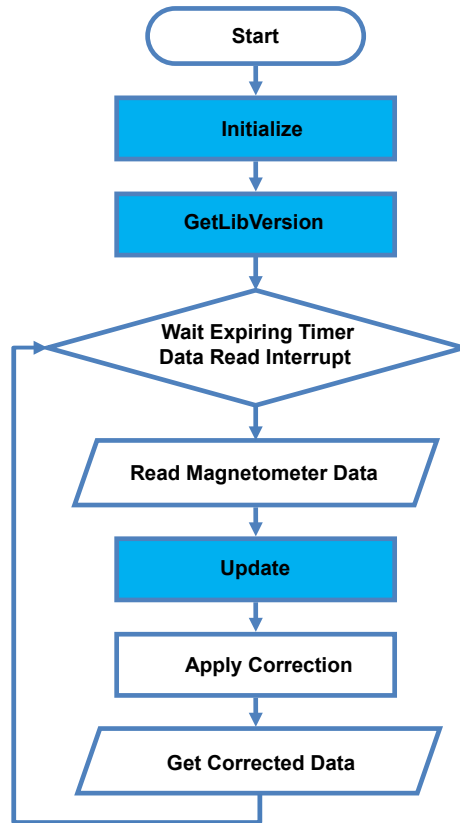
- `mcu_type` is the type of MCU:
 - `MMC_CM0P_MCU_STM32` is a standard STM32 MCU
 - `MMC_CM0P_MCU_BLUE_NRG1` is [BlueNRG-1](#)
 - `MMC_CM0P_MCU_BLUE_NRG2` is [BlueNRG-2](#)
 - `MMC_CM0P_MCU_BLUE_NRG_LP` is [BlueNRG-LP](#)
- `*mmc_algo` is a pointer to the instance of MotionMC algorithm
- `mode` parameter defines calibration mode type
 - `MMC_CM0P_HI_ONLY` hard-iron only mode (faster)
 - `MMC_CM0P_HI_AND_SI` hard-iron + soft-iron (slower)
- `enable` parameter enables (1) or disables (0) the library
- the CRC module in STM32 microcontroller (in RCC peripheral clock enable register) has to be enabled before using the library
- `void MotionMC_CM0P_Update(void *mmc_algo, MMC_CM0P_Input_t *data_in)`
 - runs the magnetometer calibration algorithm
 - `*mmc_algo` is a pointer to the instance of MotionMC algorithm
 - `*data_in` parameter is a pointer to a structure with input data
 - the parameters for the structure type `MMC_CM0P_Input_t` are:
 - `DTime` parameter is the interval between update function calls in seconds
 - `Mag[3]` is an array of magnetometer sensor value in μT

Note: *This function has to be called periodically at the same period that is indicated in the `DTime` parameter.*

- `void MotionMC_CM0P_GetCalParams (MMC_CM0P_Output_t *data_out)`
 - retrieves the magnetometer hard iron (HI) and soft iron (SI) coefficients
 - `*data_out` parameter is a pointer to a structure with output data
 - the parameters for the structure type `MMC_CM0P_Output_t` are:
 - `HI_Bias[3]` is an array of magnetometer hard iron (HI) coefficients in μT
 - `SI_Matrix[3][3]` is a 3x3 matrix of magnetometer soft iron (SI) coefficients
 - `B` is the estimated magnetic field in μT (to be used in `MMC_CM0P_HI_ONLY` mode)
 - `CalQuality` is the calibration quality factor.

2.2.3 API flow chart

Figure 1. MotionMC API logic sequence



2.2.4 Demo code

The following demonstration code reads data from magnetometer sensor and calculates compensated data.

```

[...]

#define VERSION_STR LENG 35
#define SAMPLE_TIME 20 /* Set sample timer overflow to 20 ms
                        (i.e.: 50 Hz) */

[...]

/** Initialization */
volatile uint32_t timestamp = 0; /* Increments when sample timer
                                overflows */

char lib_version[VERSION_STR LENG];

/* Magnetometer calibration API initialization function */
MotionMC_Initialize(SAMPLE_TIME, 1);

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

[...]

/** Using magnetometer calibration algorithm */
Timer_overflow_Interrupt_Handler()
{
  MMC_input_t data_in;
  MMC_output_t data_out;
  float mag_cal_x, mag_cal_y, mag_cal_z;

  /* Get magnetic field X/Y/Z in [uT] */
  MEMS_Read_MagValue(&data_in.Mag[0], &data_in.Mag[1], &data_in.Mag[2]);

  /* Get current sample time in [ms] */
  data_in.TimeStamp = timestamp * SAMPLE_TIME;

  /* Magnetometer calibration algorithm update */
  MotionMC_Update(&data_in);

  /* Get the magnetometer calibration coefficients */
  MotionMC_GetCalParams(&data_out);

  /* Apply calibration coefficients */
  mag_cal_x = (int)((data_in.Mag[0] - data_out.HI_Bias[0]) * SF_Matrix[0][0]
                  + (data_in.Mag[1] - data_out.HI_Bias[1]) * SF_Matrix[0][1]
                  + (data_in.Mag[2] - data_out.HI_Bias[2]) * SF_Matrix[0][2]);

  mag_cal_y = (int)((data_in.Mag[0] - data_out.HI_Bias[0]) * SF_Matrix[1][0]
                  + (data_in.Mag[1] - data_out.HI_Bias[1]) * SF_Matrix[1][1]
                  + (data_in.Mag[2] - data_out.HI_Bias[2]) * SF_Matrix[1][2]);

  mag_cal_z = (int)((data_in.Mag[0] - data_out.HI_Bias[0]) * SF_Matrix[2][0]
                  + (data_in.Mag[1] - data_out.HI_Bias[1]) * SF_Matrix[2][1]
                  + (data_in.Mag[2] - data_out.HI_Bias[2]) * SF_Matrix[2][2]);
}

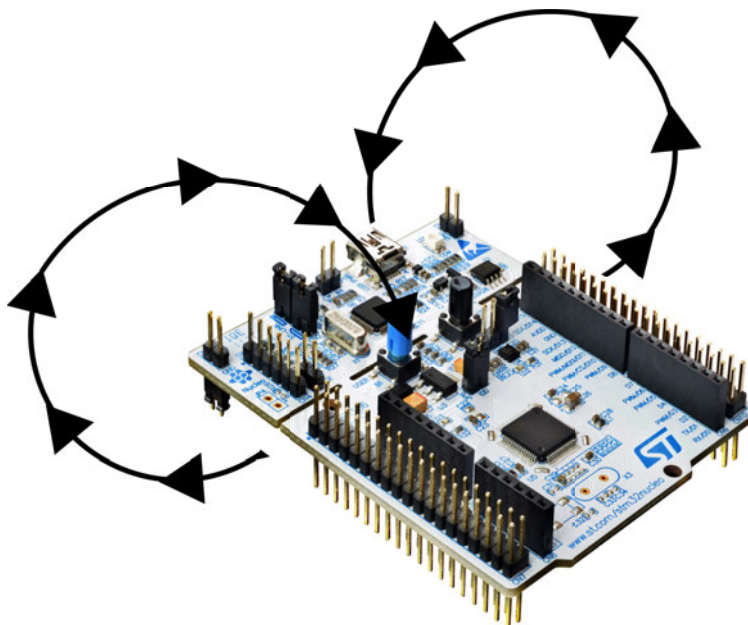
```

2.2.5 Calibration process

This calibration algorithm uses the specific motion of the magnetometer sensor exposed to Earth magnetic field.

- Step 1.** Rotate the STM32 Nucleo board slowly in a figure eight pattern through 3D space. This is not a planar movement; it is necessary to tilt and rotate the device to cover as much different 3D space positions as possible.

Figure 2. STM32 Nucleo board rotation during calibration



- Step 2.** Use the 3D Plot (see Section 3.1 [Unicleo-GUI application](#)) during calibration; the measured data points should cover as much of the 3D sphere as possible.

Important:

While performing the calibration pattern, keep the STM32Nucleo board clear of other magnetic objects such as cell phones, computers and other steel objects.

2.2.6 Algorithm performance

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

Cortex-M4 STM32F401RE at 84 MHz									Cortex-M3 STM32L152RE at 32 MHz								
STM32CubeIDE 1.2.0			IAR EWARM 8.32.3			Keil µVision 5.27			STM32CubeIDE 1.2.0			IAR EWARM 8.32.3			Keil µVision 5.27		
Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
6	16	1637	6	8	103	6	8	103	86	132	5009	80	132	1619	81	127	2101

Table 3. Cortex-M0+ and Cortex-M7: elapsed time (µs) algorithm

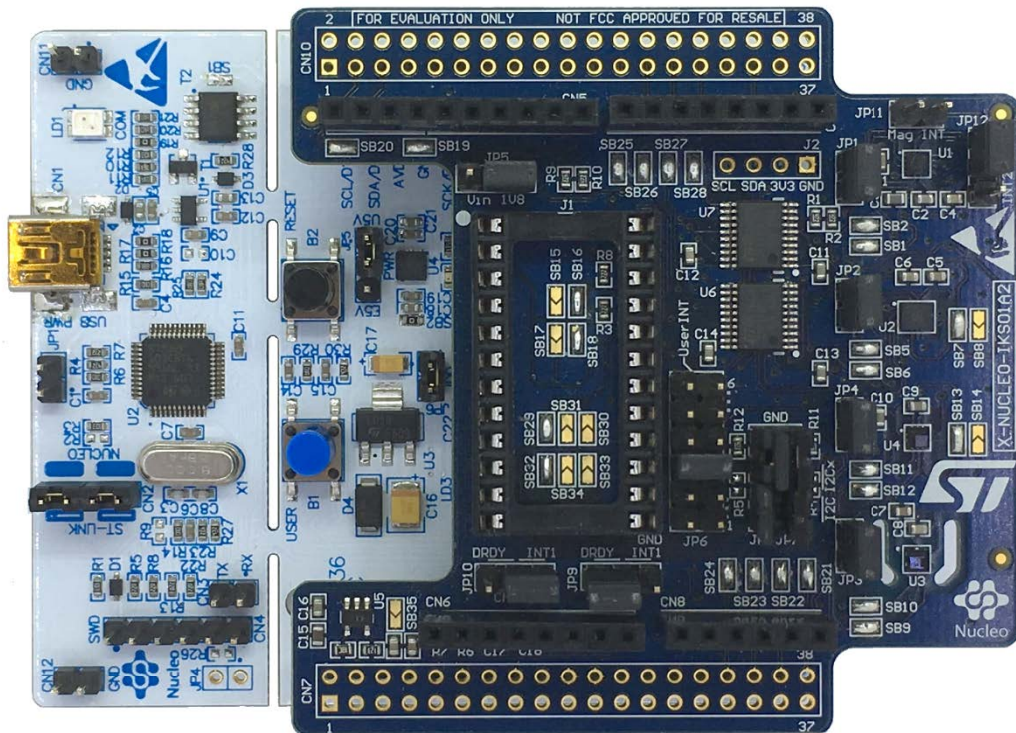
Cortex-M0+ STM32F073RZ at 32 MHz									Cortex-M7 STM32F767ZI at 96 MHz								
STM32CubeIDE 1.2.0			IAR EWARM 8.32.3			Keil µVision 5.27			STM32CubeIDE 1.2.0			IAR EWARM 8.32.3			Keil µVision 5.27		
Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
450	471	475	209	220	471	216	235	240	2	5	146	2	4	91	2	4	96

3 Sample application

The MotionMC 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, NUCLEO-L152RE or NUCLEO-L073RZ development board connected to an X-NUCLEO-IKS01A2, X-NUCLEO-IKS01A3 or X-NUCLEO-IKS02A1 expansion board.

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



Magnetometer algorithm output data may be displayed in real-time through a specific GUI.

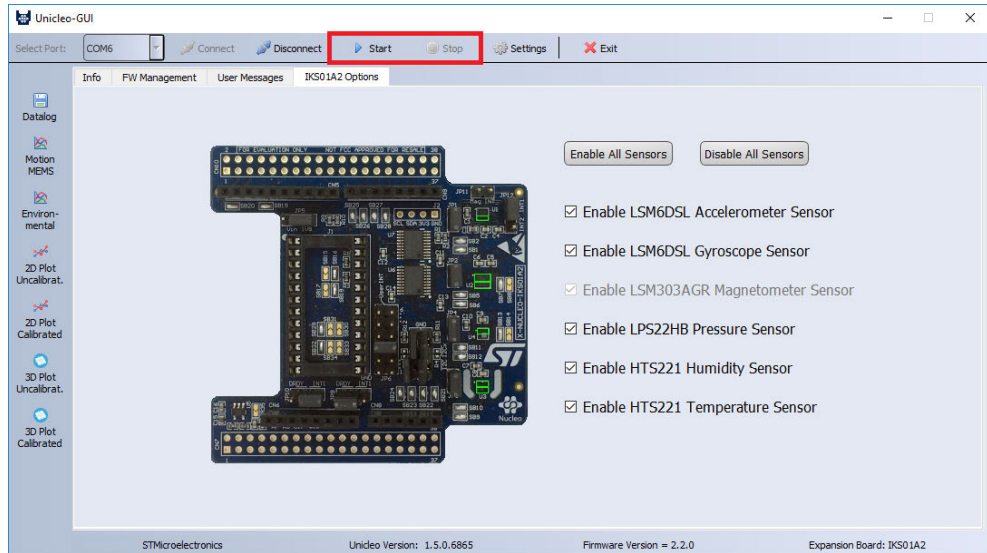
3.1 Unicleo-GUI application

The sample application uses the Windows Unicleo-GUI utility, which can be downloaded from www.st.com (refer to Section 4 References).

- Step 1.** Ensure that the necessary drivers are installed and the STM32 Nucleo board with appropriate expansion board is connected to the PC.

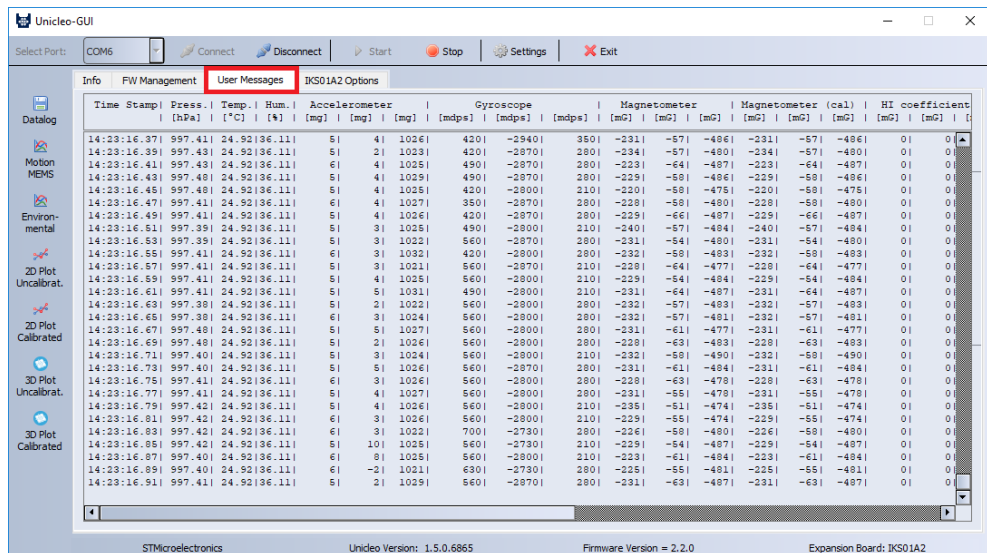
- Step 2.** Launch the Unicleo-GUI application to open the main application window.
If an STM32 Nucleo board with supported firmware is connected to the PC, it is automatically detected and the appropriate COM port is opened.

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

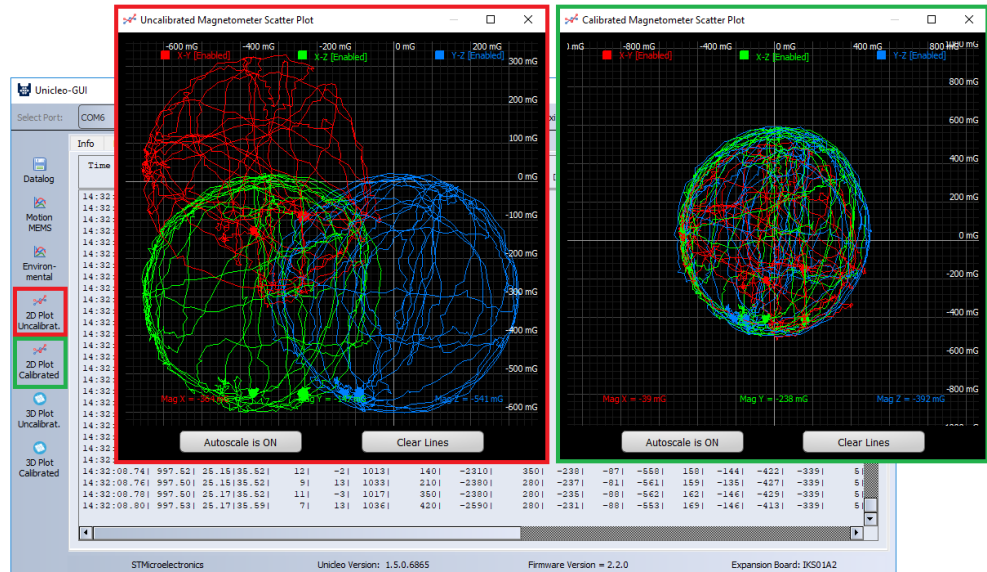


Step 4. Click on 2D Plot Uncalibrated and 2D Plot Calibrated icons in the vertical tool bar to open the corresponding windows.

The uncalibrated data window shows the hard-iron distortion. The calibrated data window shows all the data centered on the origin (0, 0, 0) as the hard-iron distortion has been removed through calibration.

- red = X-Y plane
- green = X-Z plane
- blue = Y-Z plane

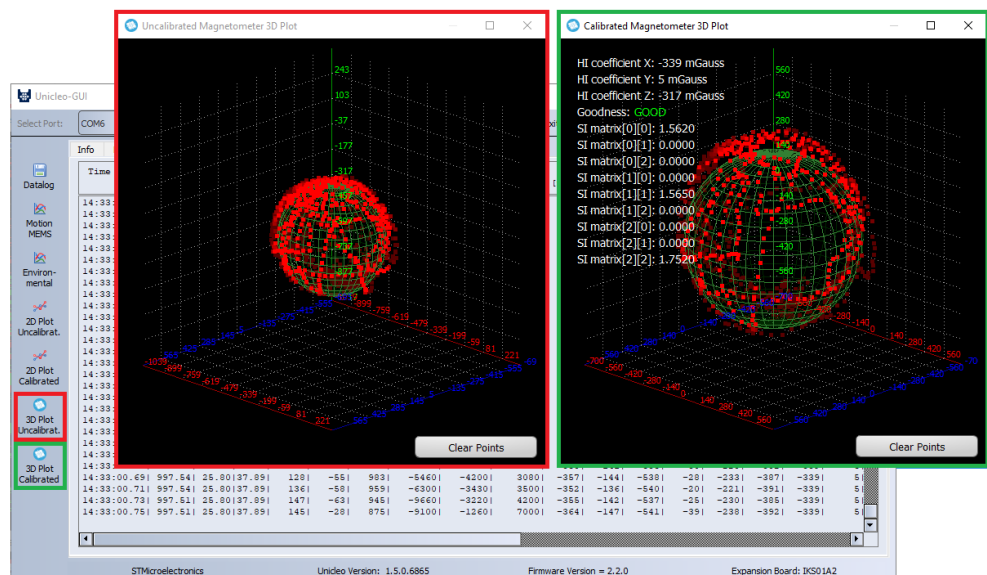
Figure 6. 2D Plot Uncalibrated/Calibrated windows



Step 5. Click on 3D Plot Uncalibrated and 3D Plot Calibrated icons in the vertical tool bar to open the corresponding windows.

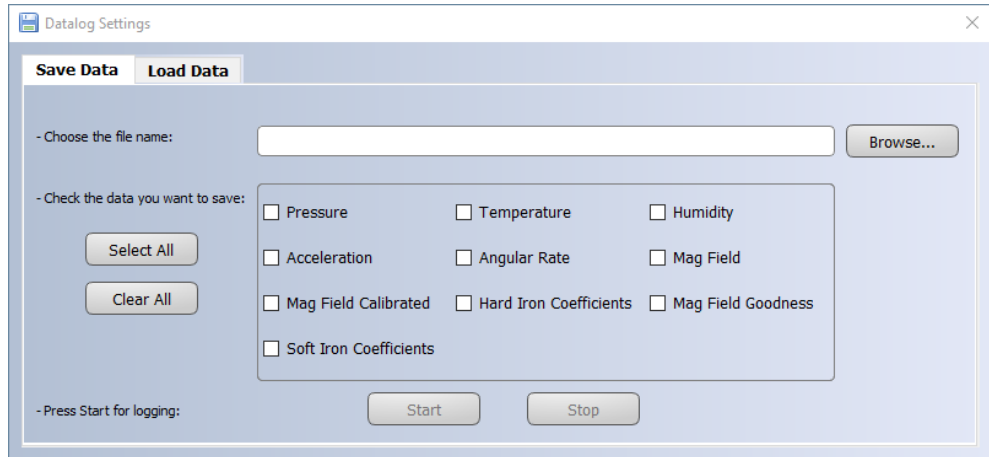
The uncalibrated data window shows the hard-iron distortion: the 3D sphere is subject to offsets in x, y and z directions. The calibrated data window shows the 3D sphere centered on the origin (0, 0, 0) as the hard-iron distortion has been removed through calibration. The Calibrated Magnetometer 3D Plot also shows the hard-iron and soft-iron coefficients and calibration quality (goodness).

Figure 7. 3D Plot Uncalibrated/Calibrated windows



- Step 6.** Click on the Datalog icon in the vertical toolbar to open the datalog configuration window: you can select the sensor and activity data to be saved in the files. You can start or stop saving by clicking on the corresponding button.

Figure 8. 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 boards (MB1136)
3. UM2128: Getting started with Unicleo-GUI for motion MEMS and environmental sensor software expansion for STM32Cube

Revision history

Table 4. Document revision history

Date	Version	Changes
10-Apr-2017	1	Initial release.
26-Jan-2018	2	Updated Section 2.1 MotionMC overview. Added references to NUCLEO-L152RE development board and Section 2.2.6 Algorithm performance.
20-Mar-2018	3	Updated Introduction and Section 2.1 MotionMC overview.
03-May-2018	4	Added Table 3. Cortex-M0+: elapsed time (μ s) algorithm.. Added references to Cortex-M0+ and NUCLEO-L073RZ development board.
21-Feb-2019	5	Updated Figure 3. Sensor expansion board and adapter connected to the STM32 Nucleo, Table 2. Cortex-M4, Cortex-M3: elapsed time (μ s) algorithm, Table 3. Cortex-M0+: elapsed time (μ s) algorithm, Figure 4. Unicleo main window, Figure 5. User Messages tab, Figure 6. 2D Plot Uncalibrated/Calibrated windows, Figure 7. 3D Plot Uncalibrated/Calibrated windows and Figure 8. Datalog window. Added X-NUCLEO-IKS01A3 expansion board compatibility information.
24-Mar-2020	6	Updated Introduction, Section 2.2.1 MotionMC library description and Section 2.2.6 Algorithm performance. Added ARM Cortex-M7 architecture compatibility information.
07-Apr-2021	7	Updated Introduction, Section 2.1 MotionMC overview , Section 2.2.1 MotionMC library description , Section 2.2.2 MotionMC APIs and Section 3 Sample application . Added X-NUCLEO-IKS02A1 compatibility information.

Contents

1	Acronyms and abbreviations	2
2	MotionMC middleware library in X-CUBE-MEMS1 software expansion for STM32Cube	3
2.1	MotionMC overview	3
2.2	MotionMC library	3
2.2.1	MotionMC library description	3
2.2.2	MotionMC APIs	4
2.2.3	API flow chart	6
2.2.4	Demo code	6
2.2.5	Calibration process	7
2.2.6	Algorithm performance	8
3	Sample application	9
3.1	Unicleo-GUI application	9
4	References	13
	Revision history	14

List of tables

Table 1.	List of acronyms	2
Table 2.	Cortex-M4, Cortex-M3: elapsed time (μ s) algorithm	8
Table 3.	Cortex-M0+ and Cortex-M7: elapsed time (μ s) algorithm	8
Table 4.	Document revision history	14

List of figures

Figure 1.	MotionMC API logic sequence	6
Figure 2.	STM32 Nucleo board rotation during calibration	8
Figure 3.	Sensor expansion board and adapter connected to the STM32 Nucleo	9
Figure 4.	Unicleo main window.	10
Figure 5.	User Messages tab	10
Figure 6.	2D Plot Uncalibrated/Calibrated windows	11
Figure 7.	3D Plot Uncalibrated/Calibrated windows	11
Figure 8.	Datalog window	12

IMPORTANT NOTICE – PLEASE 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 acknowledgement.

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, please 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.

© 2021 STMicroelectronics – All rights reserved