Getting started with MotionGC gyroscope calibration library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionGC is a middleware library part of X-CUBE-MEMS1 software and runs on STM32. It provides real-time gyroscope calibration through angular zero-rate level coefficients (offset) used to correct gyroscope 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+, 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-IKS4A1, X-NUCLEO-IKS01A3 or X-NUCLEO-IKS02A1 expansion board on a NUCLEO-F401RE, NUCLEO-U575ZI-Q, NUCLEO-L152RE or NUCLEO-L073RZ development board.

UM2181 - Rev 7 - July 2025
For further information, contact your local STMicroelectronics sales office.

www.st.com

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

2.1 MotionGC overview

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

The gyroscope sensor can have significant offset which can cause problems when using the gyroscope output data. The MotionGC library is able to minimize the offset and solve this issue.

The library acquires data from the accelerometer and gyroscope and calculates the angular zero-rate level coefficient (offset) for each axis. The angular zero-rate level coefficients are subsequently used to compensate raw data coming from gyroscope.

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 X-NUCLEO-IKS4A1, X-NUCLEO-IKS01A3 and X-NUCLEO-IKS02A1 expansion boards, mounted on a NUCLEO-F401RE, NUCLEO-U575ZI-Q, NUCLEO-L152RE or NUCLEO-L073RZ development boards.

2.2 MotionGC library

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

2.2.1 MotionGC library description

The MotionGC gyroscope calibration library manages data acquired from accelerometer and gyroscope; it features:

- the angular zero-rate level (offset) compensation
- adjustable thresholds to detect device in still state
- adjustable maximum angular zero-rate level to be compensated
- possibility to set previously saved coefficients as the initial values
- fast start-up option
- able to run with multiple sample rate from 25 Hz to 200 Hz
- resources requirements:
  – Cortex-M0+: 3.0 kB of code and 0.3 kB of data memory
  – Cortex-M3: 3.0 kB of code and 0.3 kB of data memory
  – Cortex-M33: 2.4 kB of code and 0.3 kB of data memory
  – Cortex-M4: 2.8 kB of code and 0.3 kB of data memory
  – Cortex-M7: 2.3 kB of code and 0.3 kB of data memory
- available for ARM® Cortex®-M0+, ARM® Cortex®-M3, ARM® Cortex®-M33, ARM® Cortex®-M4 and ARM® Cortex®-M7 architectures

2.2.2 MotionGC APIs

The MotionGC library APIs are:

- `uint8_t MotionGC_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 MotionGC_Initialize(MGC_mcu_type_t mcu_type, float freq)`
    - performs MotionGC library initialization and setup of the internal mechanism
    - `mcu_type` is the type of MCU:
        - `MGC_MCU_STM32` is a standard STM32 MCU
        - `MGC_MCU_BLUE_NRG1` is BlueNRG-1
        - `MGC_MCU_BLUE_NRG2` is BlueNRG-2
        - `MGC_MCU_BLUE_NRG_LP` is BlueNRG -LP
        - `MGC_MCU_STM32WB0` is STM32WB0
    - `freq` parameter is the sampling frequency

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

- `void MotionGC_SetKnobs(MGC_knobs_t *knobs)`
    - sets the library knobs
    - `*knobs` parameter is a pointer to a structure with settings
    - the parameters for the structure type `MGC_knobs_t` are the following:
        - `AccThr` is the accelerometer threshold to detect steady state in g. The default value is 0.01 g. The input range is 0.003 to 0.05 g. For higher accuracy, set the value low; for platforms inherent vibration or noisy sensor, set the value high.
        - `GyroThr` is the gyroscope threshold to detect steady state in degree per second. The default value is 0.1 dps. The input range is 0.008 to 0.4 dps. For higher accuracy, set the value low.
        - `FilterConst` is the decay constant for the internal filter (from 0 to 1). Setting a higher value can cause a variation in the gyro offset but allows a quick convergence. The default value is 0.002.
        - `FastStart` set to 1 immediately starts computing the gyro offset and quickly converges to gyro offset. By setting the value to 1, it is possible that the initial gyro offset value is not accurate but converges over a period of time.
        - `MaxGyro` is the maximum expected angular zero-rate level when still in dps. The default value is 15 dps.
        - `MaxAcc` is the maximum acceleration module when still in g. The default value is 1.3 g.
- `void MotionGC_GetKnobs(MGC_knobs_t *knobs):`
    - gets the library knob setting
    - `*knobs` is a pointer to the setting structure

Note: *This function needs to be called before making changes in the settings.*

- `void MotionGC_Update(MGC_input_t *data_in, MGC_output_t *gyro_bias, int *bias_update):`
    - runs the gyroscope calibration algorithm and returns compensation parameters
    - `*data_in` parameter is a pointer to a structure with input data
    - the parameters for the structure type `MGC_input_t` are the following:
        - `Acc[3]` is an array of accelerometer sensor value in g
        - `Gyro[3]` is an array of gyroscope sensor value in dps
    - `*gyro_bias` parameter is a pointer to a structure with gyroscope angular zero-rate level
    - the parameters for the structure type `MGC_output_t` are the following:
        - `GyroBiasX` is the actual x axis gyroscope bias value in dps
        - `GyroBiasY` is the actual y axis gyroscope bias value in dps
        - `GyroBiasZ` is the actual z axis gyroscope bias value in dps
    - `*bias_update` is a pointer to an integer set to 1 if the device is in stable state and the gyroscope bias is updated (0 otherwise)

Note: *This function has to be called periodically as indicated in the initialization function.*

- `void MotionGC_GetCalParams(MGC_output_t *gyro_bias):`
  - retrieves the gyroscope compensation parameters
  - `*gyro_bias` parameter is a pointer to a structure with gyroscope angular zero-rate level
- `void MotionGC_SetCalParams(MGC_output_t *gyro_bias)`
  - sets the initial gyroscope compensation parameters, helping to quickly converge to the gyro offset right value
  - `*gyro_bias` parameter is a pointer to a structure with gyroscope angular zero-rate level

Note: *This function should be called just after* `MotionGC_Initialize`

- `void MotionGC_SetFrequency(float freq)`
  - applies the new sample frequency to the library without resetting
  - `freq` indicates the new sample frequency in Hertz

2.2.3 API flow chart

Figure 1. API flow chart diagram
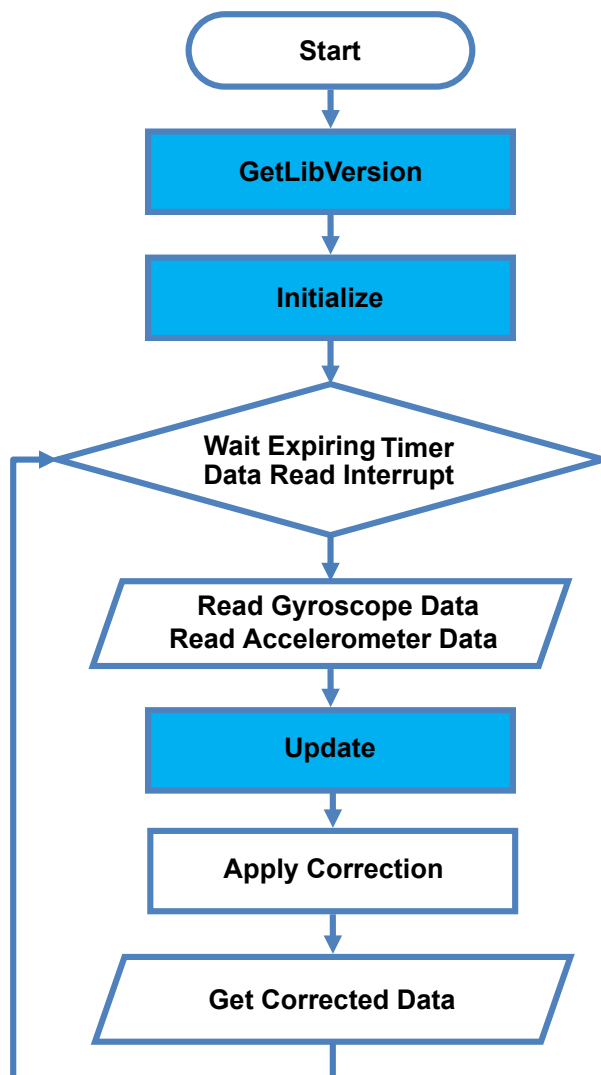
2.2.4 Demo code

The following is an example of the demonstration code that reads data from the gyroscope sensor and from the accelerometer sensor and calculates compensated data.

```
[…]

#define VERSION_STR_LENG       35
#define SAMPLE_FREQUENCY     50.0f

[…]

/* Initialization */
char lib_version[VERSION_STR_LENG];
MGC_knobs_t knobs;
MGC_output_t start_gyro_bias;
float sample_freq;

/* Gyroscope calibration API initialization function */
MotionGC_Initialize(MGC_MCU_STM32, SAMPLE_FREQUENCY);

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

/* Optional: Get knobs settings */
MotionGC_GetKnobs(&knobs);

/* Optional: Adjust knobs settings */
knobs.AccThr = 0.008f;
knobs.GyroThr = 0.15;
MotionGC_Set_knobs(&knobs);

/* Optional: Set initial gyroscope offset */
start_gyro_bias.GyroBiasX = 0;
start_gyro_bias.GyroBiasY = 0;
start_gyro_bias.GyroBiasZ = 0;
MotionGC_SetCalParams(&start_gyro_bias);

/* Optional: Set sample frequency */
sample_freq = SAMPLE_FREQUENCY;
MotionGC_SetFrequency(sample_freq);

[…]

/* Using gyroscope calibration algorithm */
Timer_OR_DataRate_Interrupt_Handler()
{
  MGC_input_t data_in;
  MGC_output_t data_out;;
  int bias_update;
  float gyro_cal_x, gyro_cal_y, gyro_cal_z;

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

  /* Get angular rate X/Y/Z in dps */
  MEMS_Read_GyroValue(&data_in.Gyro[0], &data_in.Gyro[1], &data_in.Gyro[2]);

  /* Gyroscope calibration algorithm update */
  MotionGC_Update(&data_in, &data_out, &bias_update);

  /* Apply correction */
  gyro_cal_x = (data_in.Gyro[0] – data_out.GyroBiasX);
  gyro_cal_y = (data_in.Gyro[1] – data_out.GyroBiasY);
  gyro_cal_z = (data_in.Gyro[2] – data_out.GyroBiasZ);
}
```

2.2.5 The calibration process

The calibration process does not need any sensor motion as the offset coefficient is calculated and updated when the sensor is still.

2.2.6 Algorithm performance

Table 2. Algorithm elapse time (µs) Cortex-M4, Cortex-M3 and Cortex-M0+

Cortex-M4 STM32F401RE at 84 MHz			Cortex-M3 STM32L152RE at 32 MHz			Cortex-M0+ STM32L073RZ at 32 MHz		
Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
3	7	10	10	120	211	<1	240	1000

Table 3. Algorithm elapse time (µs) Cortex-M33 and Cortex-M7

Cortex- M33 STM32U575ZI-Q at 160 MHz			Cortex- M7 STM32F767ZI at 96 MHz		
Min	Avg	Max	Min	Avg	Max
1	3	4	4	7	14

2.3 Sample application

The MotionGC 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-U575ZI-Q, NUCLEO-L152RE or NUCLEO-L073RZ development boards connected to an X-NUCLEO-IKS4A1, X-NUCLEO-IKS01A3 or X-NUCLEO-IKS02A1 expansion board.

Figure 2. STM32 Nucleo: LEDs, button, jumper
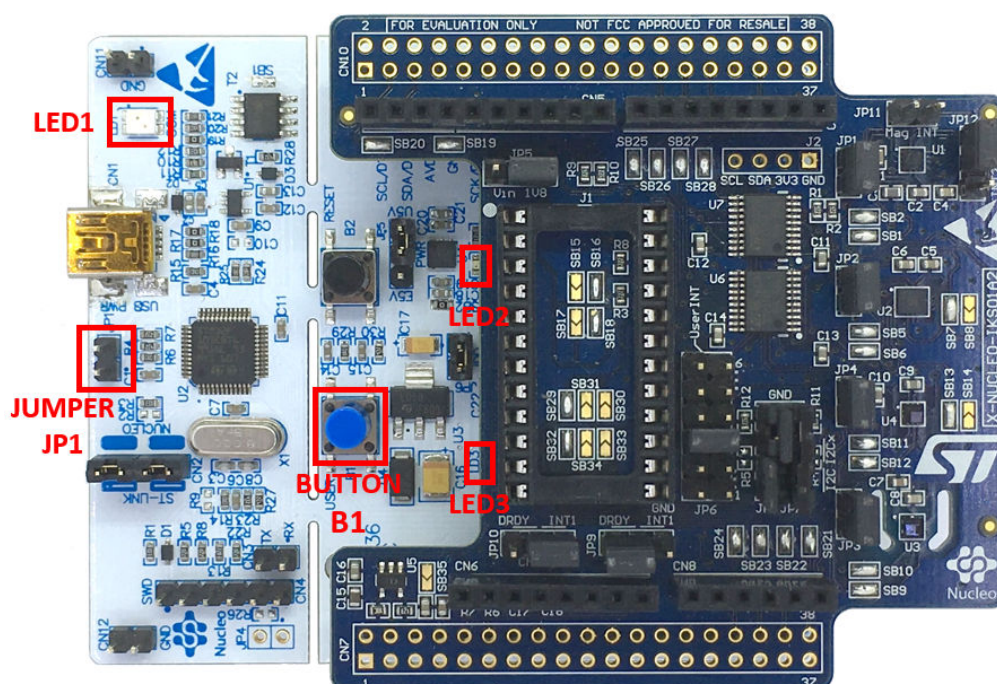
The above figure shows the user button B1 and the three LEDs of the NUCLEO-F401RE board. Once the board is powered, LED LD3 (PWR) turns ON.

A USB cable connection is required to monitor real-time data. The board is powered by the PC via USB connection. This working mode allows the user to test and display the gyroscope algorithm output data using the MEMS-Studio.

2.3.1 MEMS Studio application

The sample application uses the Windows MEMS-Studio utility, which can be downloaded from www.st.com.

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

Step 2. Launch the MEMS-Studio application to open the main application window. If an STM32 Nucleo board with the supported firmware is connected to the PC, it is automatically detected. Press the [**Connect**] button to establish connection to the evaluation board.

Figure 3. MEMS-Studio - Connect
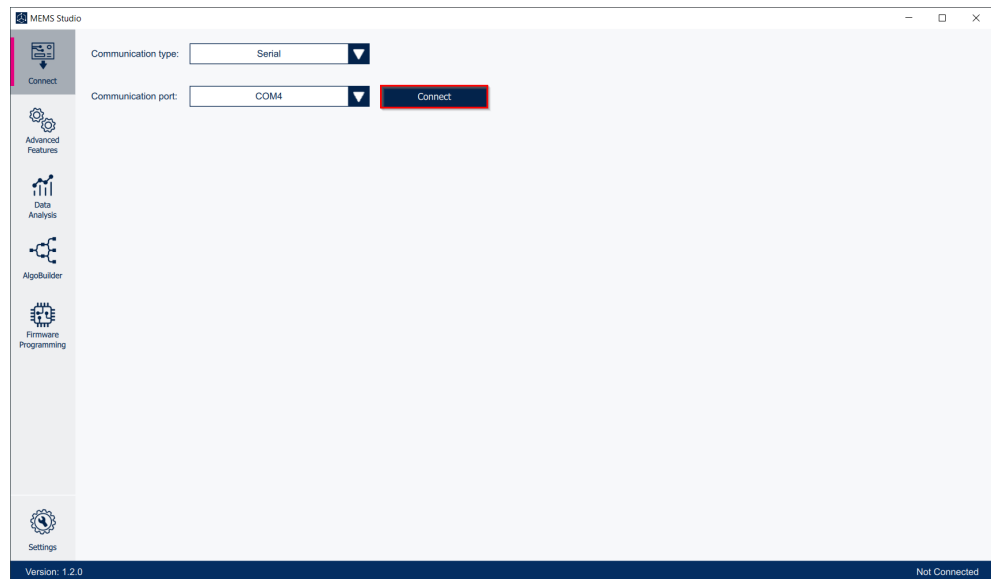
Step 3. When connected to a STM32 Nucleo board with supported firmware [**Library Evaluation**] tab is opened.

To start and stop data streaming, toggle the appropriate [**Start**] or [**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 4. User Messages tab
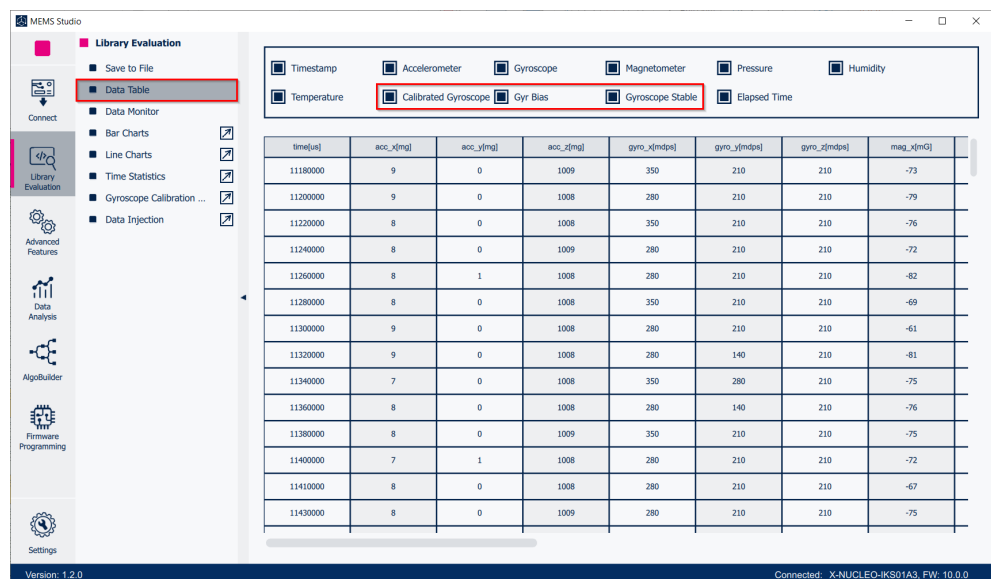
Step 4. Click on the [**Gyroscope Calibration**] to open the dedicated application window. The window is split in three sections: the first one containing uncalibrated data, the second one containing calibrated data and the last one containing offset and stable flag information.

**Figure 5. MEMS-Studio - Library Evaluation - Gyroscope Calibration**

Step 5. Click on the [**Save To File**] to open the dataloging configuration window. Select the sensor and gyroscope calibration data to be saved in the file. You can start or stop saving by clicking on the corresponding button.

Figure 6. MEMS-Studio - Library Evaluation - Save To File
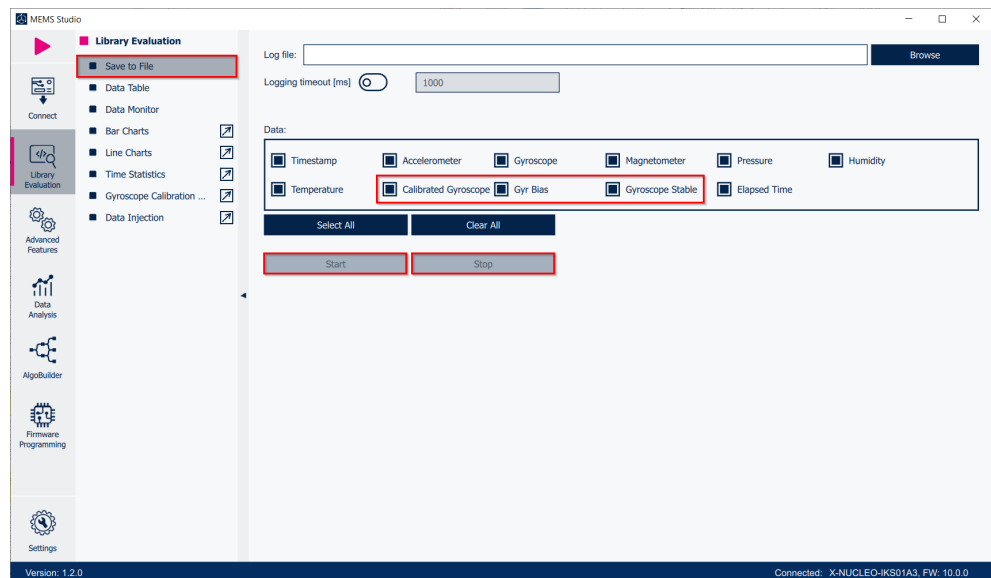
Step 6. Data Injection mode can be used to send the previously acquired data to the library and receive the result. Select the [**Data Injection**] tab on the vertical tool bar to open the dedicated view for this functionality.

Figure 7. MEMS-Studio - Library Evaluation - Data Injection
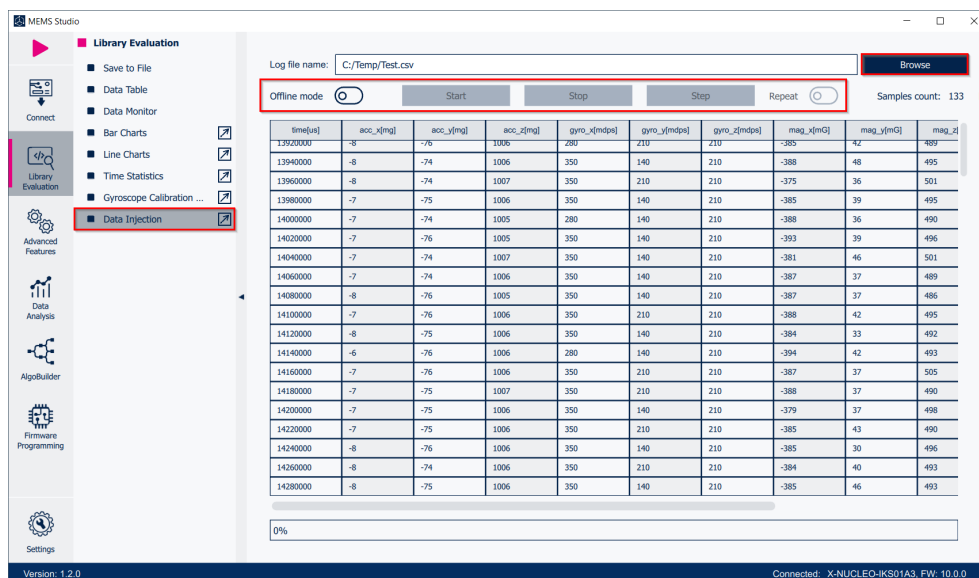
Step 7. Click on the [**Browse**] button to select the file with the previously captured data in CSV format.

The data will be loaded into the table in the current view.
Other buttons will become active. You can click on:

– [**Offline Mode**] button to switch the firmware offline mode on/off (mode utilizing the previously captured data).

– [**Start**]/[**Stop**]/[**Step**]/[**Repeat**] buttons to control the data feed from MEMS-Studio to the library.

3 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
03-Mar-2017	1	Initial release.
26-Jan-2018	2	Added references to NUCLEO-L152RE development board and Section 2.2.6 Algorithm performance.
20-Mar-2018	3	Updated Introduction and Section 2.1 MotionGC overview.
02-May-2018	4	Updated Section 2.2.1 MotionGC library description and Table 2. Cortex-M4 and Cortex-M3: elapsed time (µs) algorithm..
		Added Table 3. Cortex-M0+: elapsed time (µs) algorithm.
		Added references to ARM Cortex-M0+ and NUCLEO-L073RZ development board.
19-Feb-2019	5	Updated Table 2. Cortex-M4 and Cortex-M3: elapsed time (µs) algorithm, Table 3. Cortex-M0+: elapsed time (µs) algorithm, Figure 2. Sensor expansion board and adapter connected to an STM32 Nucleo, Figure 3. Unicleo main window, Figure 4. User Messages tab, Figure 5. Gyroscope calibration window and Figure 6. Datalog window.
		Added X-NUCLEO-IKS01A3 expansion board compatibility information.
23-Mar-2020	6	Updated Introduction, *Section 2.2.1: MotionGC library description* and *Section 2.2.6: Algorithm performance.*
		Added ARM Cortex-M7 architecture compatibility information.
01-Jul-2025	7	Updated Section Introduction, Section 2.1: MotionGC overview, Section 2.2.1: MotionGC library description, Section 2.2.2: MotionGC APIs, Section 2.2.4: Demo code, Section 2.2.6: Algorithm performance, Section 2.3: Sample application, Section 2.3.1: MEMS Studio application.

Contents

List of tables

List of figures

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.