User manual

Getting started with MotionCP real-time carry position library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionCP middleware library is part of the X-CUBE-MEMS1 software and runs on STM32. It provides real-time information about how the user is carrying a device (i.e. cell phone).

It is able to distinguish the following positions: on desk, in hand, near head, shirt pocket, trouser pocket, swinging arm and jacket pocket.

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 to ease portability across different STM32 microcontrollers.

The software comes with sample implementation running on X-NUCLEO-IKS4A1 or X-NUCLEO-IKS01A3 expansion board on a NUCLEO-F401RE, NUCLEO-U575ZI-Q 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 MotionCP middleware library in X-CUBE-MEMS1 software expansion for STM32Cube

2.1 MotionCP overview

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

The library acquires data from the accelerometer and provides information about how the user is carrying the 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 for X-NUCLEO-IKS4A1 and X-NUCLEO-IKS01A3 expansion boards, mounted on a NUCLEO-F401RE, NUCLEO-U575ZI-Q or NUCLEO-L152RE development board.

2.2 MotionCP library

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

2.2.1 MotionCP library description

The MotionCP carry position recognition library manages the data acquired from the accelerometer; it features:

- possibility to distinguish the following positions: on desk, in hand, near head, shirt pocket, trouser pocket, arm swing, jacket pocket
- recognition based on the accelerometer data only
- required accelerometer data sampling frequency of 50 Hz
- resources requirements:
  - Cortex-M3: 6.8 kB of code and 11.0 kB of data memory
  - Cortex-M33: 6.9 kB of code and 11.0 kB of data memory
  - Cortex-M4: 6.8 kB of code and 11.0 kB of data memory
  - Cortex-M7:7.0 kB of code and 11.0 kB of data memory
- available for ARM® Cortex®-M3, ARM® Cortex®-M33, ARM® Cortex®-M4 and ARM® Cortex®-M7 architectures

2.2.2 MotionCP APIs

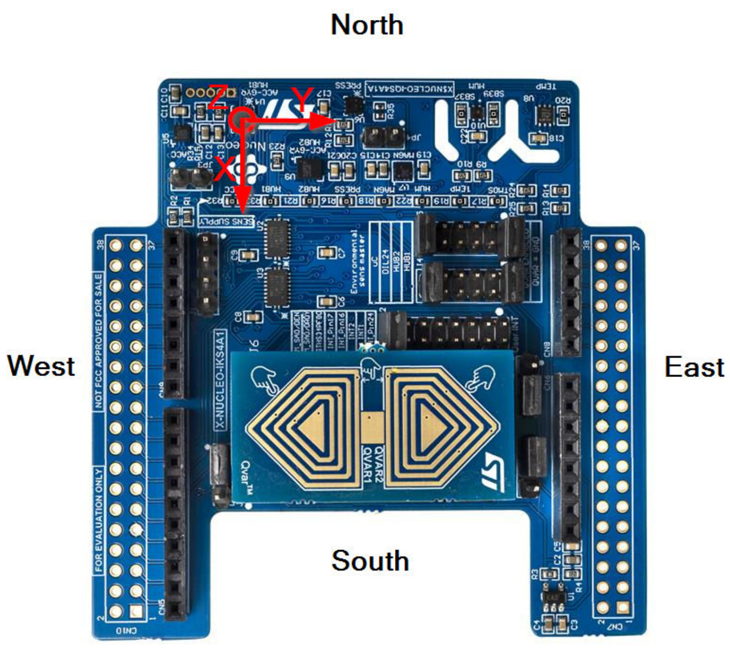
The MotionPE library APIs are:

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

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

- `void MotionCP_Update(MCP_input_t *data_in, MCP_output_t *data_out)`
  - executes carry position algorithm
  - `*data_in` parameter is a pointer to a structure with input data
  - the parameters for the structure type `MCP_input_t` are:
    - `AccX` is the accelerometer sensor value in X axis in g
    - `AccY` is the accelerometer sensor value in Y axis in g
    - `AccZ` is the accelerometer sensor value in Z axis in g
  - `*data_out` parameter is a pointer to an enum with the following items:
    - `MPE_UNKNOWN = 0`
    - `MCP_ONDESK = 1`
    - `MCP_INHAND = 2`
    - `MCP_NEARHEAD = 3`
    - `MCP_SHIRTPOCKET = 4`
    - `MCP_TROUSERPOCKET = 5`
    - `MCP_ARMSWING = 6`
    - `MCP_JACKETPOCKET = 7`
- `void MotionCP_SetOrientation_Acc(const char *acc_orientation)`
  - this function is used to set the accelerometer data orientation
  - configuration is usually performed immediately after the `MotionCP_Initialize` function call
  - `*acc_orientation` parameter is a pointer to a string of three characters indicating the direction of each of the positive orientations of the reference frame used for accelerometer data output, in the sequence x, y, z. Valid values are: n (north) or s (south), w (west) or e (east), u (up) or d (down).
    As shown in the figure below, the X-NUCLEO-IKS4A1 accelerometer sensor has an SEU orientation (x - South, y - East, z - Up), so the string is: "seu".

Figure 1. Example of sensor orientations



2.2.3 API flow chart

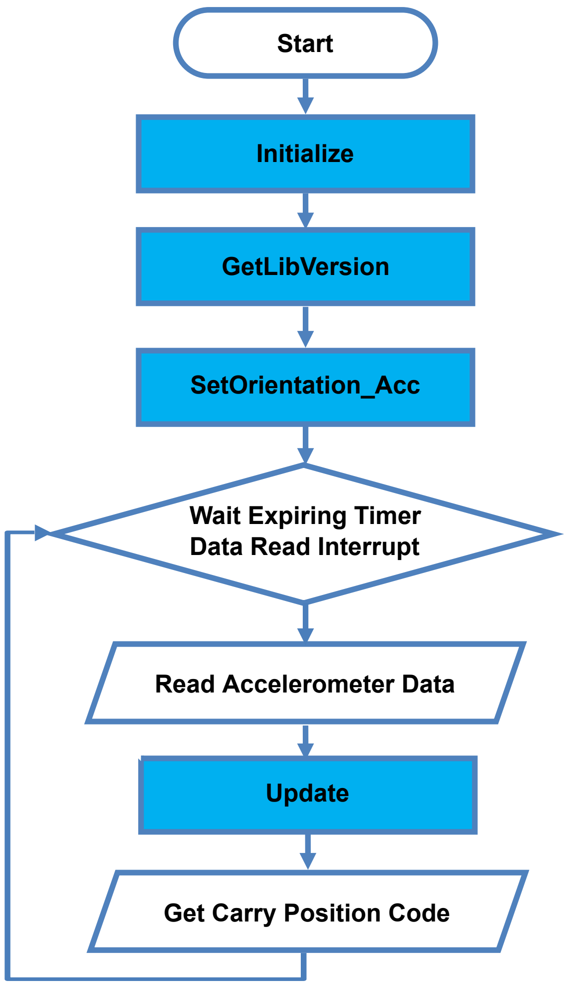


Figure 2. MotionCP API logic sequence

2.2.4 Demo code

The following demonstration code reads data from the accelerometer sensor and gets the carry position code.

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

/* Initialization */
char lib_version[VERSION_STR_LENG];
char acc_orientation[3];

/* Carry position API initialization function */
MotionCP_Initialize();

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

/* Set accelerometer orientation */
acc_orientation[0] ='n';
```

```
acc_orientation[1] ='w';
acc_orientation[2] ='u';
MotionCP_SetOrientation_Acc(acc_orientation);

[…]

/* Using Carry Position algorithm */
Timer_OR_DataRate_Interrupt_Handler()
{
  MCP_input_t data_in;
  MCP_output_t data_out;

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

  /* Carry Position algorithm update */
  MotionCP_Update(&data_in, &data_out);
}
```

2.2.5 Algorithm performance

The carry position recognition algorithm only uses data from the accelerometer and runs at a low frequency (50 Hz) to reduce power consumption.

The detected position is a phone typical carry position as the algorithm is sensitive to orientation, in particular for in hand and near head positions. Some other carry positions (like arm swing and trouser pocket) are only detected when the person is walking.

Note: *When replicating phone activity with the STM 32 Nucleo board, ensure the USB connector is oriented downwards, as it is on a phone (see the figure below).*

Figure 3. STM32 Nucleo vs phone orientation

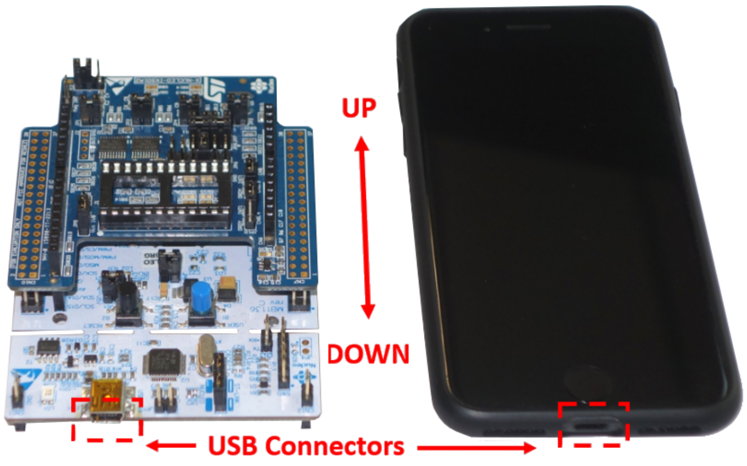


Table 2. Algorithm performance data

Carry position	Detection probability (typical)[(1)]	Best performance	Susceptible
On desk	95.31%	Normal use cases when phone is on desk	Vulnerable to sustained vibrations like banging on the desk or continuously tapping on the phone
In hand	99.31%	Correct orientation; i.e., natural phone carrying positions in hand while looking, reading or texting. Robust for stationary, walking and fast walking scenarios.	Horizontal orientation/ panorama orientation are not considered
Near head	96.31%	Correct orientation,i.e. carrying phone while talking. Robust for stationary, walking and fast walking scenarios.	Wrong orientation

Carry position	Detection probability (typical)(1)	Best performance	Susceptible
Shirt pocket	98.29%	Robust for walking and fast walking scenarios.	For stationary scenarios, the torso posture determines the algorithm performance
Trouser pocket	98.68%	Robust for walking scenarios, for both front and back trouser pockets and multiple orientation in which the phone can be carried while it is in in the trouser pocket.	Stationary
Swinging arm	97.68%	Walking	Stationary
Jacket pocket	94.73%	Walking	Stationary

1. *Typical specifications are not guaranteed.*

Typical detection latency is 5 second.

Table 3. Algorithm elapse time (µs) Cortex-M4, Cortex-M3

Cortex-M4 STM32F401RE at 84 MHz			Cortex-M3 STM32L152RE at 32 MHz		
Min	Avg	Max	Min	Avg	Max
141	190	2780	502	680	9887

Table 4. Algorithm elapse time (µs) Cortex-M33 and Cortex-M7

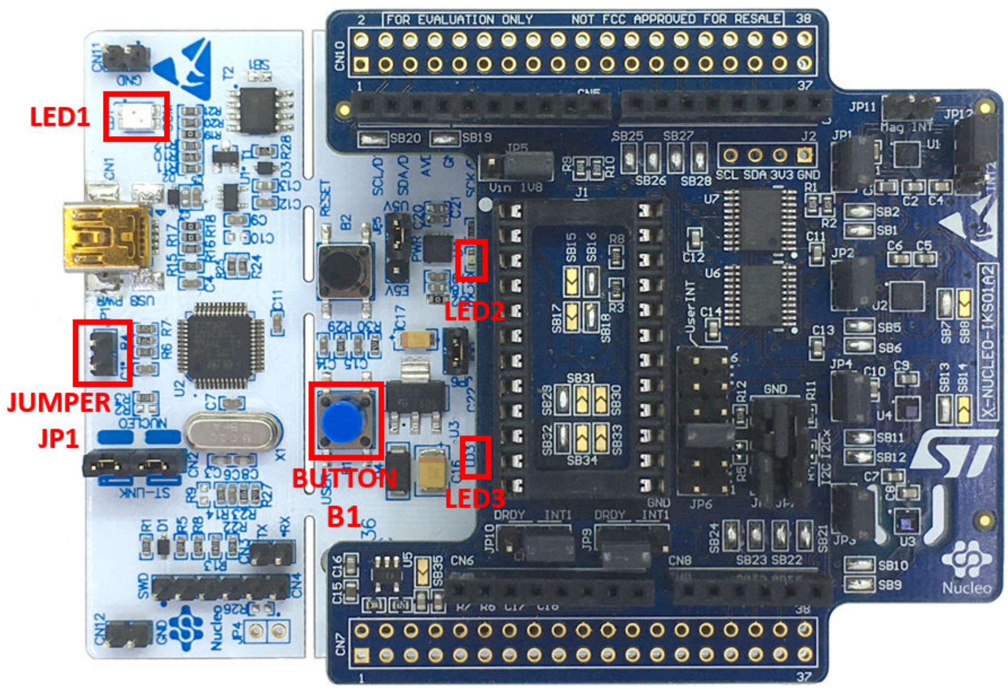
Cortex- M33 STM32U575ZI-Q at 160 MHz			Cortex- M7 STM32F767ZI at 96 MHz		
Min	Avg	Max	Min	Avg	Max
70	97	1484	267	391	6277

2.3 Sample application

The MotionCP 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 or NUCLEO-L152RE development board connected to an X-NUCLEO-IKS4A1 or X-NUCLEO-IKS01A3 expansion board.

Figure 4. **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 display real-time carry positions, accelerometer data, time stamp and eventually other sensor data, using the MEMS-Studio.

2.4 MEMS Studio application

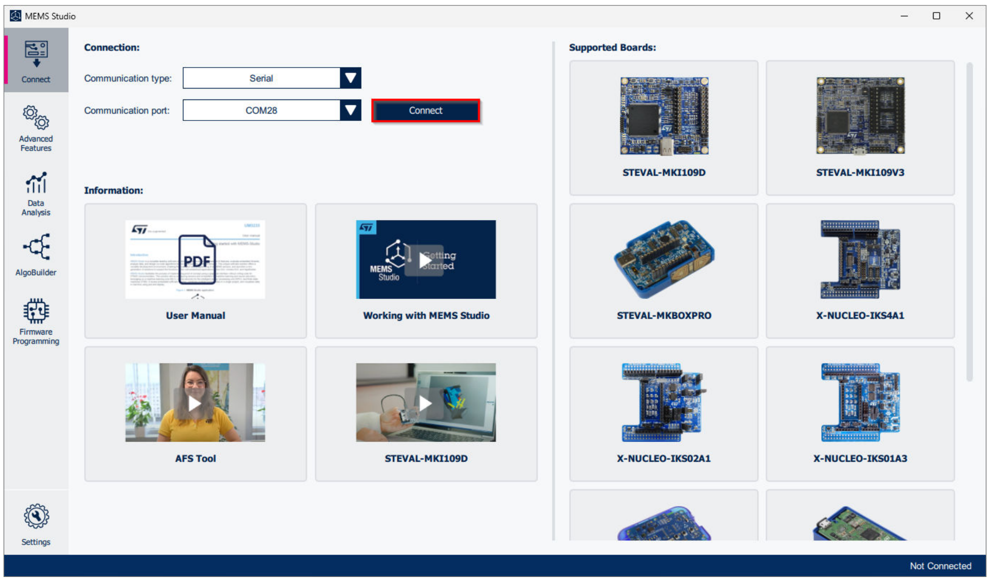
The sample application uses the Windows MEMS-Studio utility, which can be downloaded from www.st.com.

Step 1. Ensure that the necessary drivers are installed and the STM32 Nucleo board with 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 supported firmware is connected to the PC, it is automatically detected. Press the [**Connect**] button to establish connection to the evaluation board.

Figure 5. **MEMS-Studio - Connect**



Communication type: Serial

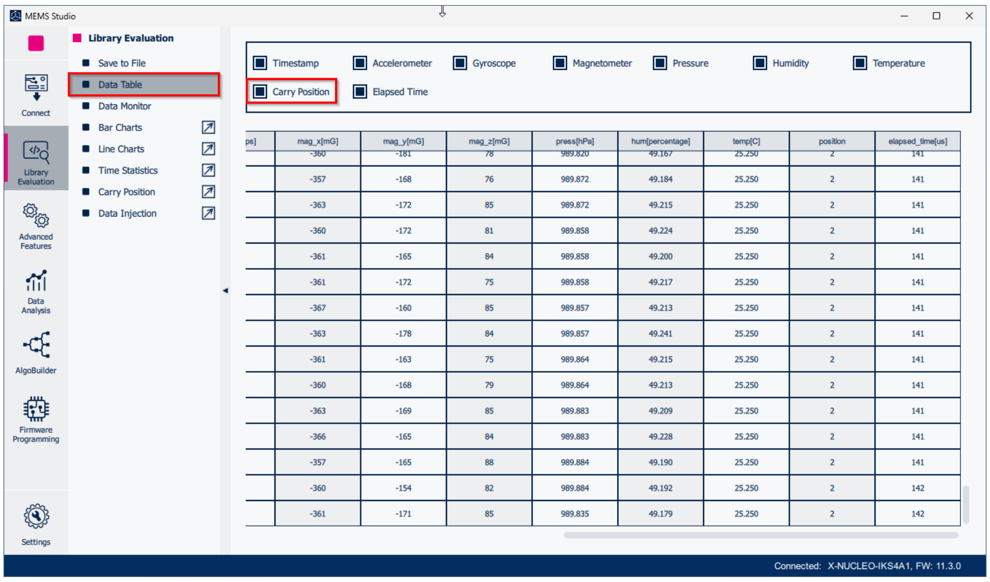
Communication port: COM28

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 6. **MEMS-Studio - Library Evaluation - Data Table**

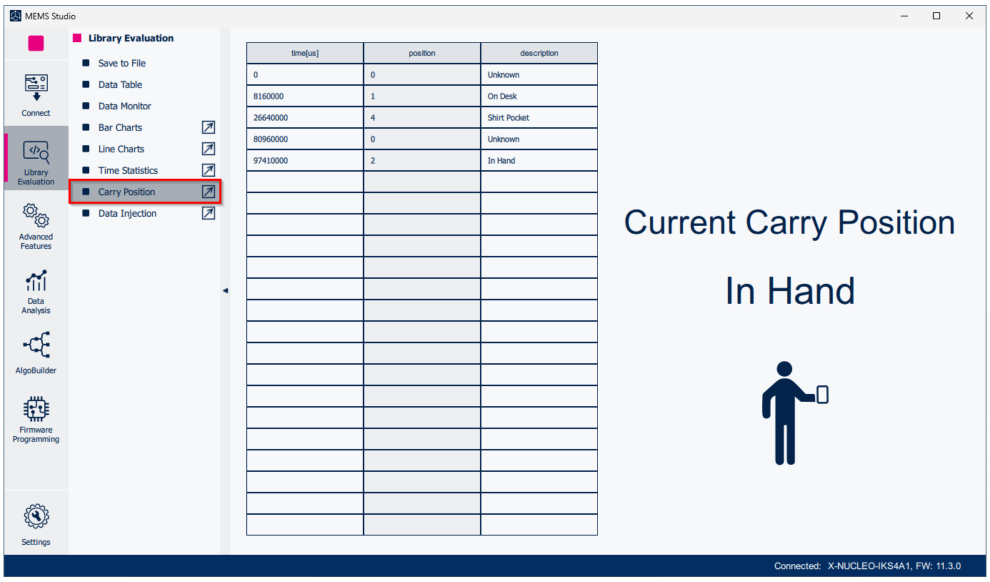


ps]	mag_x[mG]	mag_y[mG]	mag_z[mG]	press[hPa]	hum[percentage]	temp[C]	position	elapsed_time[us]
	-360	-181	78	989.820	49.167	25.250	2	141
	-357	-168	76	989.872	49.184	25.250	2	141
	-363	-172	85	989.872	49.215	25.250	2	141
	-360	-172	81	989.858	49.224	25.250	2	141
	-361	-165	84	989.858	49.200	25.250	2	141
	-361	-172	75	989.858	49.217	25.250	2	141
	-367	-160	85	989.857	49.213	25.250	2	141
	-363	-178	84	989.857	49.241	25.250	2	141
	-361	-163	75	989.864	49.215	25.250	2	141
	-360	-168	79	989.864	49.213	25.250	2	141
	-363	-169	85	989.883	49.209	25.250	2	141
	-366	-165	84	989.883	49.228	25.250	2	141
	-357	-165	88	989.884	49.190	25.250	2	141
	-360	-154	82	989.884	49.192	25.250	2	142
	-361	-171	85	989.835	49.179	25.250	2	142

Connected: X-NUCLEO-IKS4A1, FW: 11.3.0

Step 4. Click on the [**Carry Position**] to open the dedicated application window.

Figure 7. MEMS-Studio - Library Evaluation - Carry Position



time[us]	position	description
0	0	Unknown
8160000	1	On Desk
26640000	4	Shirt Pocket
80960000	0	Unknown
97410000	2	In Hand

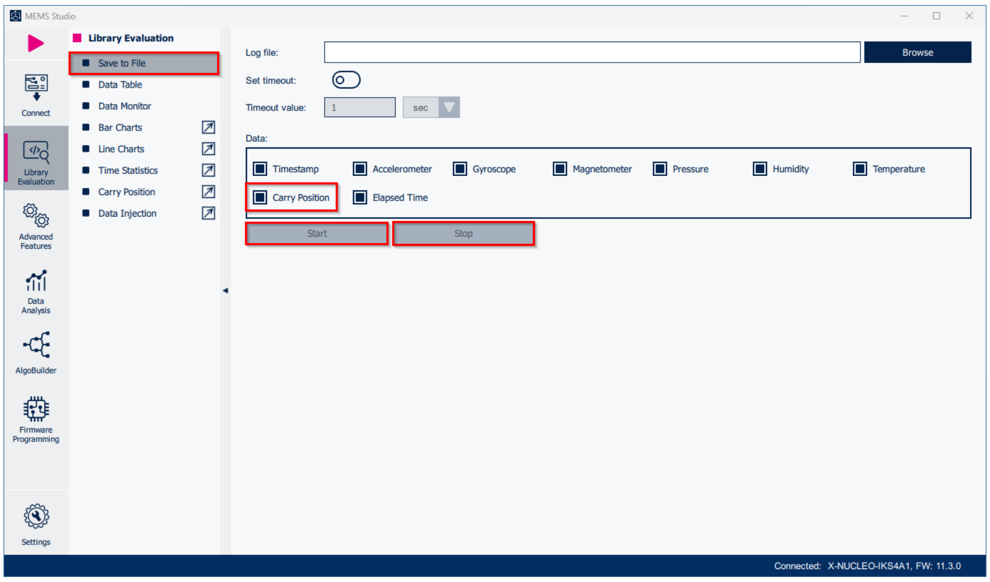
Current Carry Position

In Hand

Connected: X-NUCLEO-IKS4A1, FW: 11.3.0

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

Figure 8. MEMS-Studio - Library Evaluation - Save To File



Log file:

Set timeout:

Timeout value: 1 sec

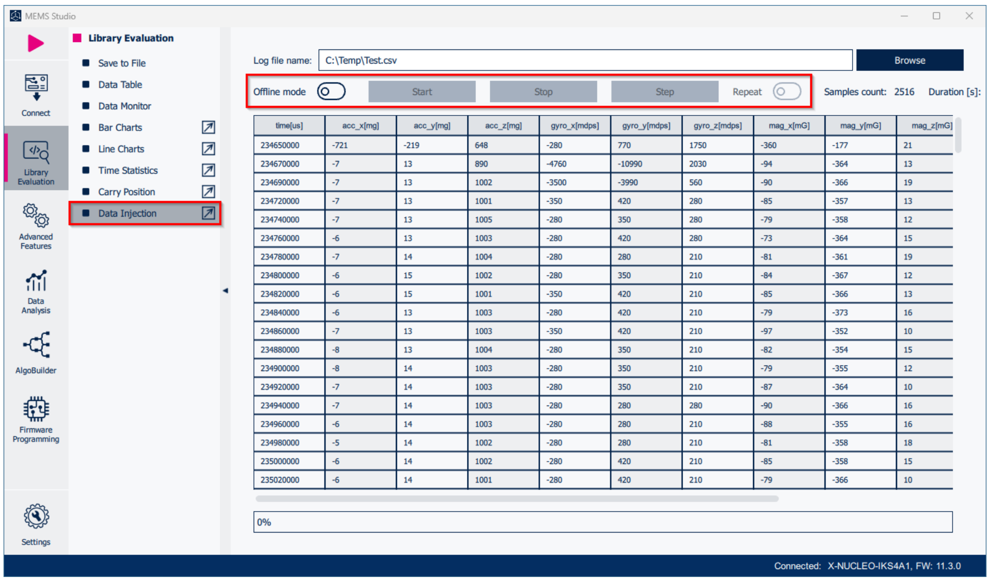
Data:

- [✓] Timestamp
- [✓] Accelerometer
- [✓] Gyroscope
- [✓] Magnetometer
- [✓] Pressure
- [✓] Humidity
- [✓] Temperature
- [✓] Carry Position
- [✓] Elapsed Time

Connected: X-NUCLEO-IKS4A1, FW: 11.3.0

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 9. **MEMS-Studio - Library Evaluation - Data Injection**



Log file name: C:\Temp\Test.csv

Samples count: 2516 Duration [s]:

time[us]	acc_x[mg]	acc_y[mg]	acc_z[mg]	gyro_x[mdps]	gyro_y[mdps]	gyro_z[mdps]	mag_x[mG]	mag_y[mG]	mag_z[mG]
234650000	-721	-219	648	-280	770	1750	-360	-177	21
234670000	-7	13	890	-4760	-10990	2030	-94	-364	13
234690000	-7	13	1002	-3500	-3990	560	-90	-366	19
234720000	-7	13	1001	-350	420	280	-85	-357	13
234740000	-7	13	1005	-280	350	280	-79	-358	12
234760000	-6	13	1003	-280	420	280	-73	-364	15
234780000	-7	14	1004	-280	280	210	-81	-361	19
234800000	-6	15	1002	-280	350	210	-84	-367	12
234820000	-6	15	1001	-350	420	210	-85	-366	13
234840000	-6	13	1003	-280	420	210	-79	-373	16
234860000	-7	13	1003	-350	420	210	-97	-352	10
234880000	-8	13	1004	-280	350	210	-82	-354	15
234900000	-8	14	1003	-280	350	210	-79	-355	12
234920000	-7	14	1003	-280	350	210	-87	-364	10
234940000	-7	14	1003	-280	280	280	-90	-366	16
234960000	-6	14	1003	-280	280	210	-88	-355	16
234980000	-5	14	1002	-280	280	210	-81	-358	18
235000000	-6	14	1002	-280	420	210	-85	-358	15
235020000	-6	14	1001	-280	420	210	-79	-366	10

0%

Connected: X-NUCLEO-IKS4A1, FW: 11.3.0

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 5. Document revision history

Date	Version	Changes
15-May-2017	1	Initial release.
25-Jan-2018	2	Added references to NUCLEO-L152RE development board and Table 3. Elapsed time (µs) algorithm.
20-Mar-2018	3	Updated Introduction, Section 2.1 MotionCP overview and Section 2.2.5 Algorithm performance.
18-Feb-2019	4	Updated Figure 1. Example of sensor orientations, Figure 3. STM32 Nucleo vs phone orientation, Table 3. Elapsed time (µs) algorithm and Figure 4. STM32 Nucleo: LEDs, button, jumper. Added X-NUCLEO-IKS01A3 expansion board compatiblity information.
25-Mar-2020	5	Updated Introduction, Section 2.2.1: MotionCP library description and Section 2.2.5: Algorithm performance. Added ARM Cortex-M7 architecture compatibility information.
01-Jul-2025	6	Updated Section Introduction, Section 2.1: MotionCP overview, Section 2.2.1: MotionCP library description, Section 2.2.2: MotionCP APIs, Section 2.2.5: Algorithm performance, Section 2.3: Sample application, Section 2.4: MEMS Studio application

Contents

List of tables

List of figures