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 or ARM® Cortex®-M4 architecture.

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

The software comes with a 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 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-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 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
- 5.6 kByte of code memory and 12 kByte of data memory usage

  Note: Real size might differ for different IDEs (toolchain)
- available for ARM® Cortex®-M3 and ARM Cortex-M4 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
  - the CRC module in STM32 microcontroller (in RCC peripheral clock enable register) has to be enabled before using the library

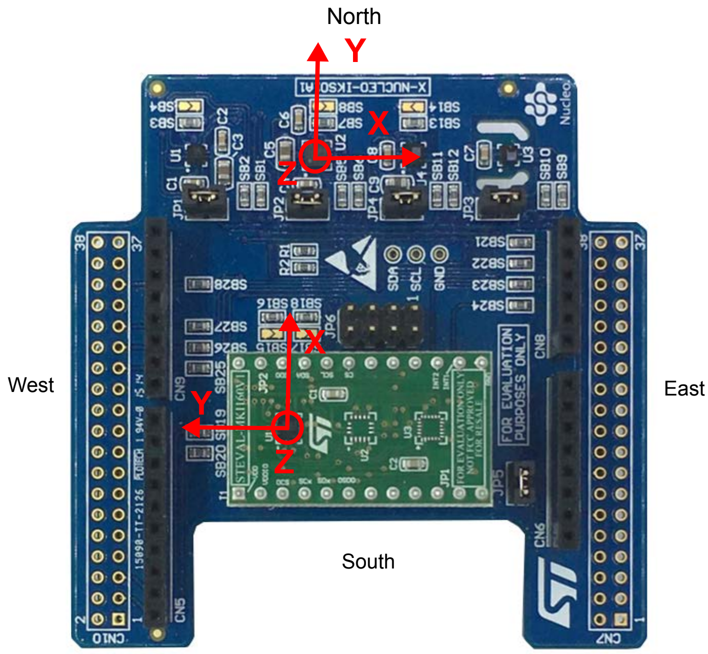
  Note: This function must be called before using the carry position library
- `void MotionCP_Update(MCP_input_t *data_in, MCP_output_t *data_out)`
  - executes carry position algorithm
  - `*data_in` parameter is a pointer to a structure with input data
  - the parameters for the structure type `MCP_input_t` 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).

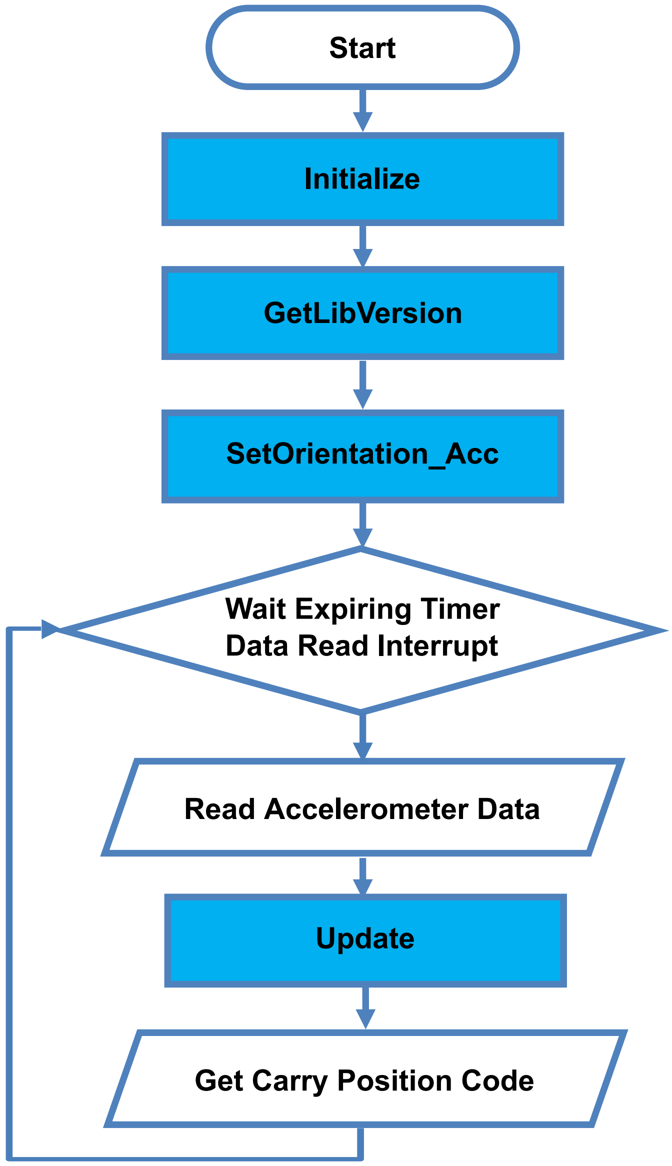
    As shown in the figure below, the X-NUCLEO-IKS01A1 accelerometer sensor has an ENU orientation (x - East, y - North, z - Up), so the string is: “enu”, while the accelerometer sensor in STEVAL-MKI160V1 is NWU (x-North, y-West, z-Up): “nwu”.

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

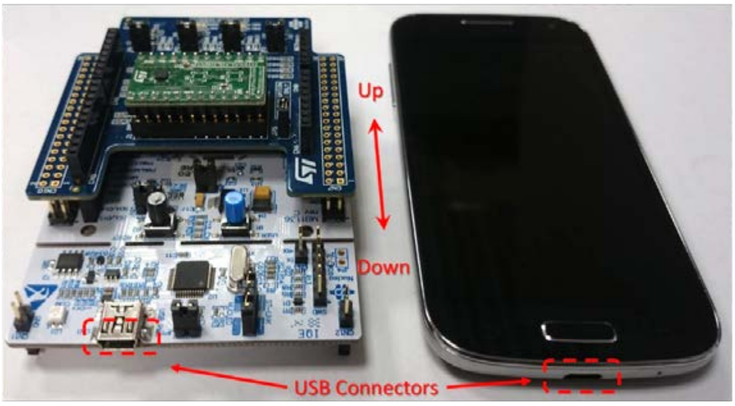


Table 2. Algorithm performance data

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
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. 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
16	228	3506	150	224	4504	146	457	6402	168	773	11297	519	754	14096	515	1420	19332

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-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).

The application recognizes the carry positions in real-time. The data can be displayed through a GUI or stored in the board for offline analysis.

The algorithm recognizes the following positions: on desk, in hand, near head, shirt pocket, trouser pocket, arm swing and jacket pocket.

Stand-alone mode

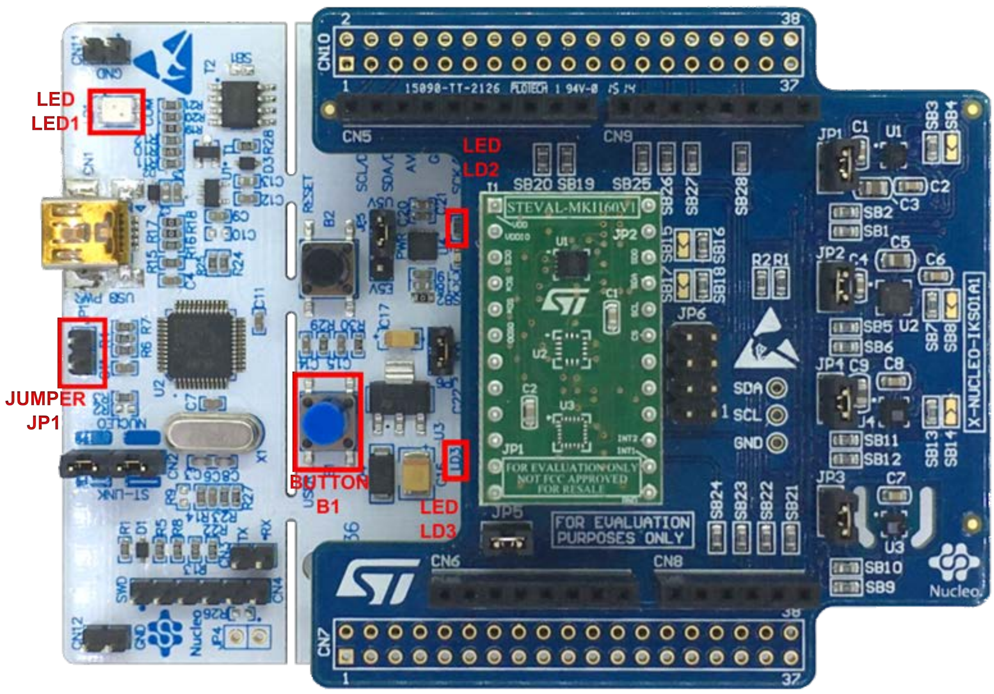
In stand-alone mode, the sample application allows the user to detect the performed gesture and store it in the MCU flash memory.

The STM32 Nucleo board may be supplied by a portable battery pack (to make the user experience more comfortable, portable and free of any PC connections).

Table 4. Power supply scheme

Power source	JP1 settings	Working mode
USB PC cable	JP1 open	PC GUI driven mode
Battery pack	JP1 closed	Stand-alone mode

Figure 4. STM32 Nucleo: LEDs, button, jumper



The above figure shows the user button B1 and the three LEDs on the NUCLEO-F401RE board. Once the board is powered, LED LD3 (PWR) turns ON and the tricolor LED LD1 (COM) begins blinking slowly due to the missing USB enumeration (refer to UM1724 on www.st.com for further details).

Note: After powering the board, LED LD2 blinks once indicating the application is ready.

When the user button B1 is pressed, the system starts acquiring data from the accelerometer sensor and detects the carry position. During this acquisition mode, fast LED LD2 blinking indicates that the algorithm is running; the detected user pose is stored in the MCU internal flash memory. Data are automatically saved every 5 minutes to avoid excessive data loss in case of an unforeseen power fault.

Pressing button B1 a second time stops the algorithm and data storage and LED LD2 switches off.

Pressing the button again starts the algorithm and data storage once again.

The flash sector dedicated to data storage is 128 KB, allowing memorization of more than 16,000 data sets.

To retrieve these data, the board must be connected to a PC running Unicleo-GUI. When stored data is retrieved via the GUI, the MCU flash sector dedicated to this purpose is cleared.

If LED LD2 is ON after powering the board, it represents a warning message indicating the flash memory is full.

Note: Optionally, the MCU memory can be erased by holding the user push button down for at least 5 seconds. LED LD2 switches OFF and then blinks 3 times to indicate that the data stored in the MCU has been erased. This option is available only after power ON or reset of the board while LED LD2 is ON indicating the flash memory is full.

When the application runs in stand-alone mode and the flash memory is full, the application switches to PC GUI drive mode and LED LD2 switches OFF.

The flash memory must be erased by downloading data via the Unicleo-GUI or the user push button (see the above note).

PC GUI drive mode

In this mode, 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 counter values, accelerometer and pressure data, time stamp and any other sensor data, using the Unicleo-GUI.

In this working mode, data are not stored in the MCU flash memory.

2.4 Unicleo-GUI application

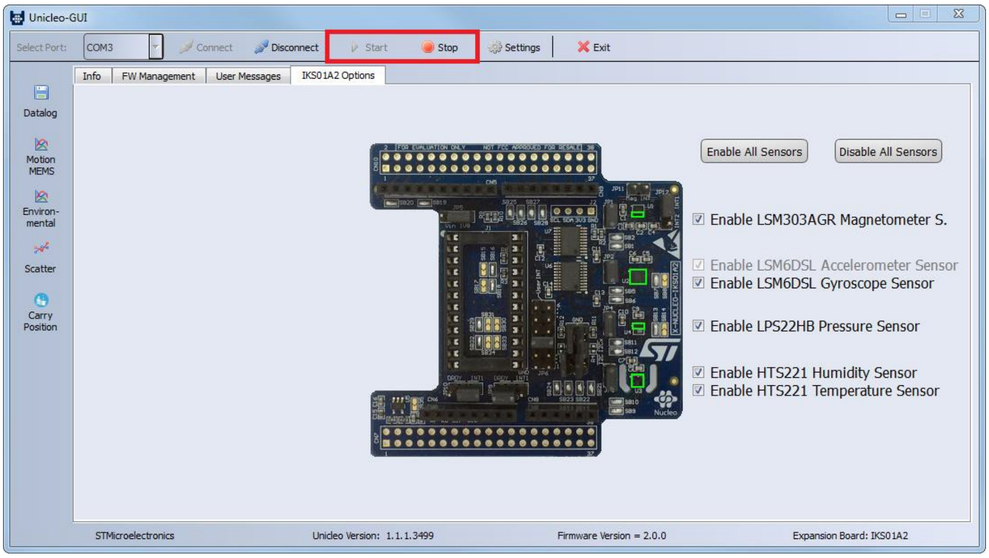
The sample application uses the Windows Unicleo-GUI utility, which can be downloaded from www.st.com.

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 5. 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.

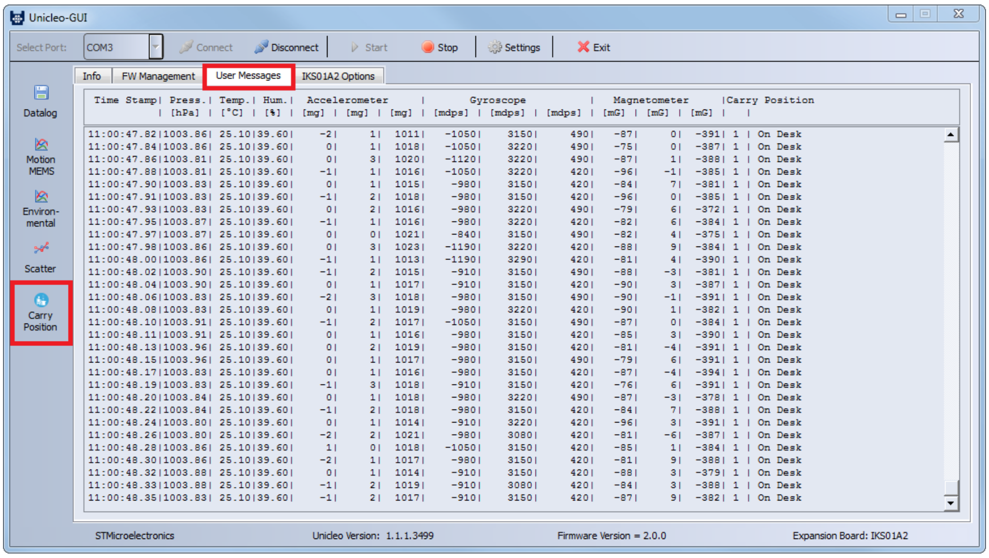


Figure 6. User Messages tab

Step 4. Click on the Carry position icon in the vertical tool bar to open the dedicated application window.

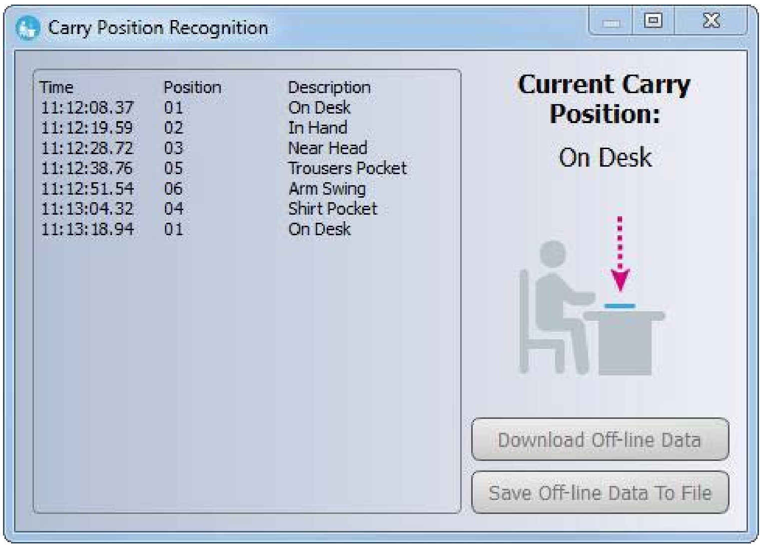


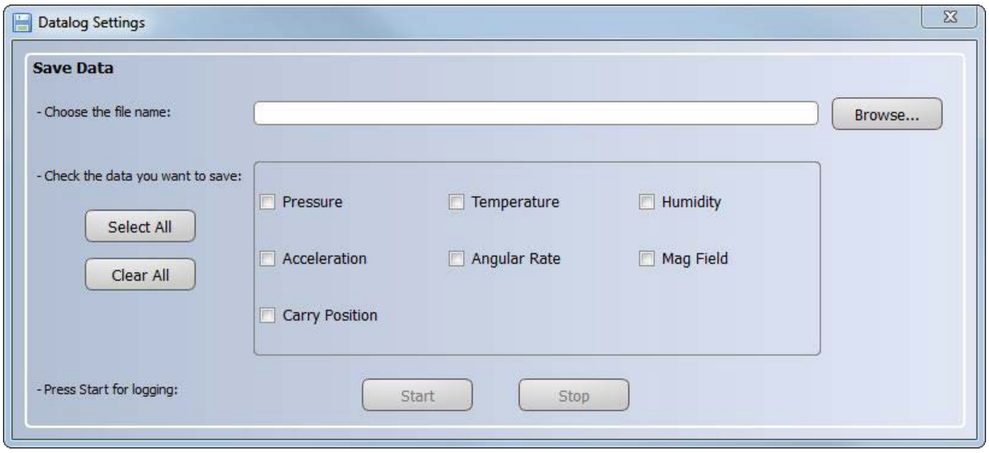
Figure 7. Carry Position window

If the board has been working in standalone mode and the user wants to retrieve stored data, press **Download Off-line Data** button to upload the stored activities data to the application. This operation automatically deletes acquired data from microcontroller.

Press the **Save Off-line Data to File** button to save the uploaded data in a .tsv file.

Step 5. Click on the Datalog icon in the vertical tool bar to open the datalog configuration window: you can select which sensor and activity data to save in files. You can start or stop saving by clicking on the corresponding button.

Figure 8. Datalog window



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

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 Section • Introduction, Section 2.1 MotionCP overview and Section 2.2.5 Algorithm performance.

Contents

List of tables

List of figures