UM2275

User manual

Getting started with MotionFD real-time fall detection library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionFD is a middleware library part of X-CUBE-MEMS1 software and runs on STM32. It provides real-time information about the user fall event based on data from a device. It is able to distinguish whether the user fall occurred or not.

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 that eases 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 MotionFD middleware library in X-CUBE-MEMS1 software expansion for STM32Cube

2.1 MotionFD overview

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

The library acquires data from the accelerometer and pressure sensor and provides information about the user fall event 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.

Sample implementation is available for the X-NUCLEO-IKS4A1 and X-NUCLEO-IKS01A3 expansion board, mounted on a NUCLEO-F401RE, NUCLEO-U575ZI-Q or NUCLEO-L152RE development board.

2.2 MotionFD library

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

2.2.1 MotionFD library description

The MotionFD fall detection library manages the data acquired from the accelerometer and pressure sensor; it features:

- possibility to distinguish whether the user fall occurred or not
- recognition based only on accelerometer and pressure sensor data
- required accelerometer and pressure sensor data sampling frequency is 25 Hz
- resources requirements:
  - Cortex-M3: 3.6 kB of code and 3.2 kB of data memory
  - Cortex-M33: 3.4 kB of code and 3.2 kB of data memory
  - Cortex-M4: 3.4 kB of code and 3.2 kB of data memory
  - Cortex-M7: 3.4 kB of code and 3.2 of data memory
- available for ARM Cortex-M3, ARM Cortex-M33, ARM Cortex-M4 and ARM Cortex-M7 architectures

2.2.2 MotionFD APIs

The MotionFD library APIs are:

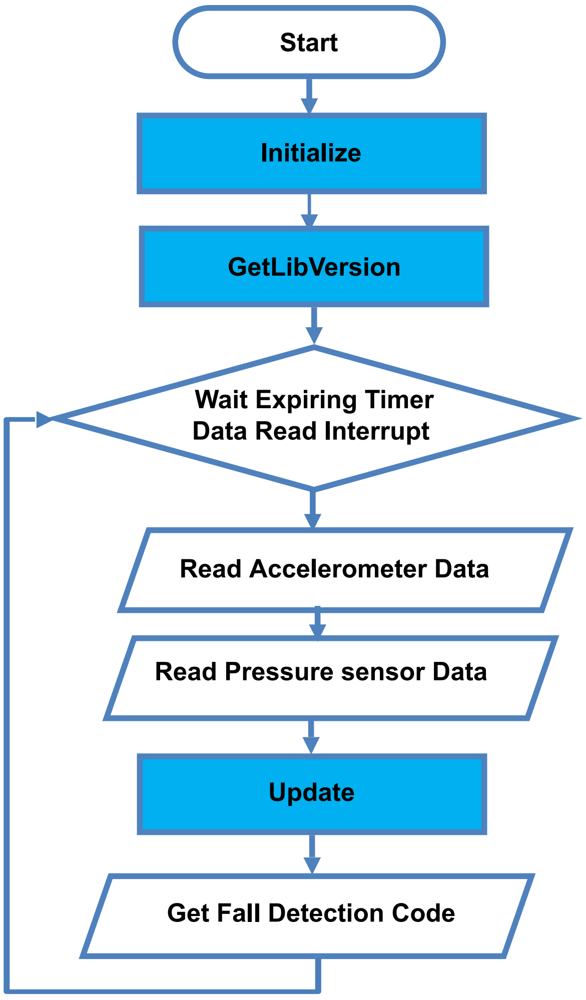
- `uint8_t MotionFD_GetLibVersion(char *version)`
  - retrieves the library version
  - `*version` is a pointer to an array of 35 characters
  - returns the number of characters in the version string
- `void MotionFD_Initialize(void)`
  - performs MotionFD library initialization and setup of the internal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

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

- `void MotionFD_Update (MFD_input_t *data_in, MFD_output_t *data_out)`
  - executes fall detection algorithm
  - `*data_in` parameter is a pointer to a structure with input data
  - the parameters for the structure type `MFD_input_t` are:
    - `AccX` is the accelerometer sensor value in X axis in mg
    - `AccY` is the accelerometer sensor value in Y axis in mg
    - `AccZ` is the accelerometer sensor value in Z axis in mg
    - `Press` is the pressure sensor value in hPa
  - `*data_out` parameter is a pointer to an enum with the following items:
    - `MFD_NOFALL = 0`
    - `MFD_FALL = 1`
- `void MotionFD_SetKnobs(float fall_threshold, int32_t fall_altitude_delta, float lying_time)`
  - sets library configuration parameters
  - `fall_threshold` acceleration threshold in mg
  - `fall_altitude_delta` altitude difference in cm
  - `lying time` time in seconds without movement after an impact
- `void MotionFD_GetKnobs(float *fall_threshold, int32_t *fall_altitude_delta, float *lying_time)`
  - gets library configuration parameters
  - `fall_threshold` acceleration threshold in mg
  - `fall_altitude_delta` altitude difference in cm
  - `lying time` time in seconds without movement after an impact

2.2.3 API flow chart

Figure 1. MotionFD API logic sequence



2.2.4 Demo code

The following demonstration code reads data from the accelerometer and pressure sensor and gets the fall event code.

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

/* Initialization */
char lib_version[VERSION_STR_LENG];

/* Fall Detection API initialization function */
MotionFD_Initialize();

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

[…]

/* Using Fall Detection algorithm */
Timer_OR_DataRate_Interrupt_Handler()
{
  MFD_input_t data_in;
  MFD_output_t data_out;

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

  /* Get pressure in hPa */
  MEMS_Read_PressValue(&data_in.Press);

  /* Fall Detection algorithm update */
  MotionFD_Update(&data_in, &data_out);
}
```

2.2.5 Algorithm performance

The fall detection algorithm only uses data from the accelerometer and pressure sensor and runs at a low frequency (25 Hz) to reduce power consumption.

Table 2. **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
62	66	105	218	260	639

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
32	34	54	20	31	83

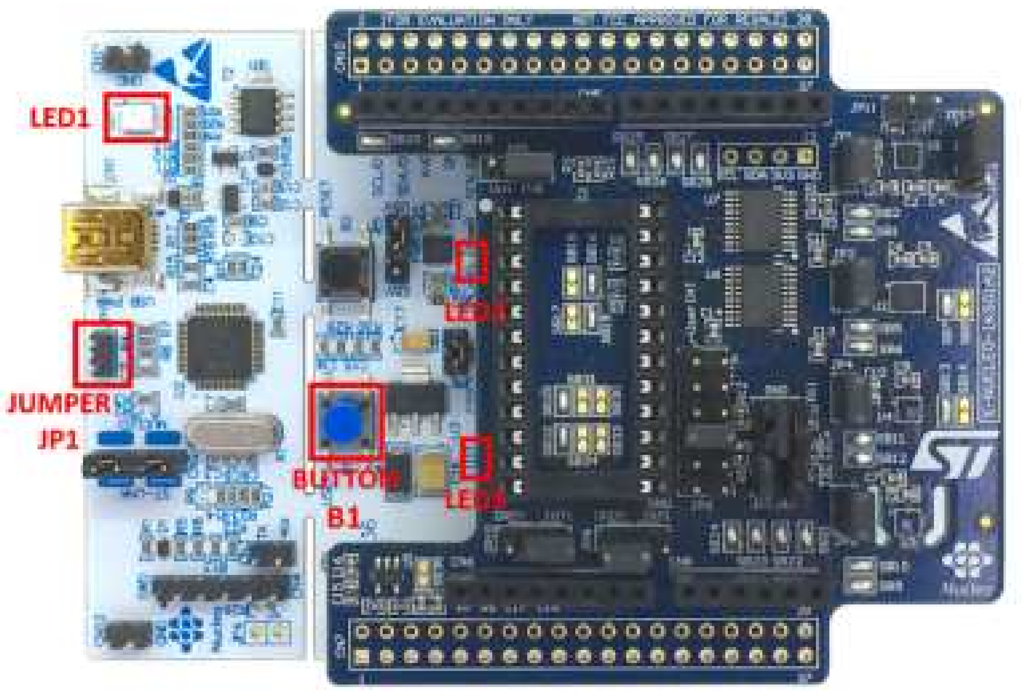
2.3 Sample application

The MotionFD 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.

The application recognizes the user fall event in real-time.

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 detected user fall event, accelerometer and pressure sensor data, time stamp and eventually other sensor data, in real-time, using the MEMS-Studio.

2.4 MEMS-Studio application

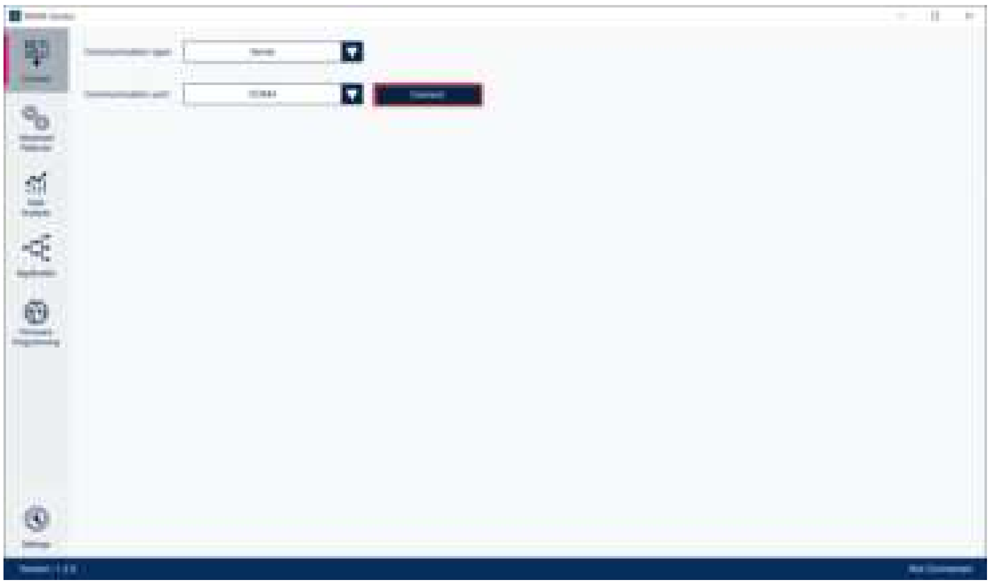
The sample application uses MEMS-Studio application, which can be downloaded from www.st.com.

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 3. **MEMS-Studio - Connect**

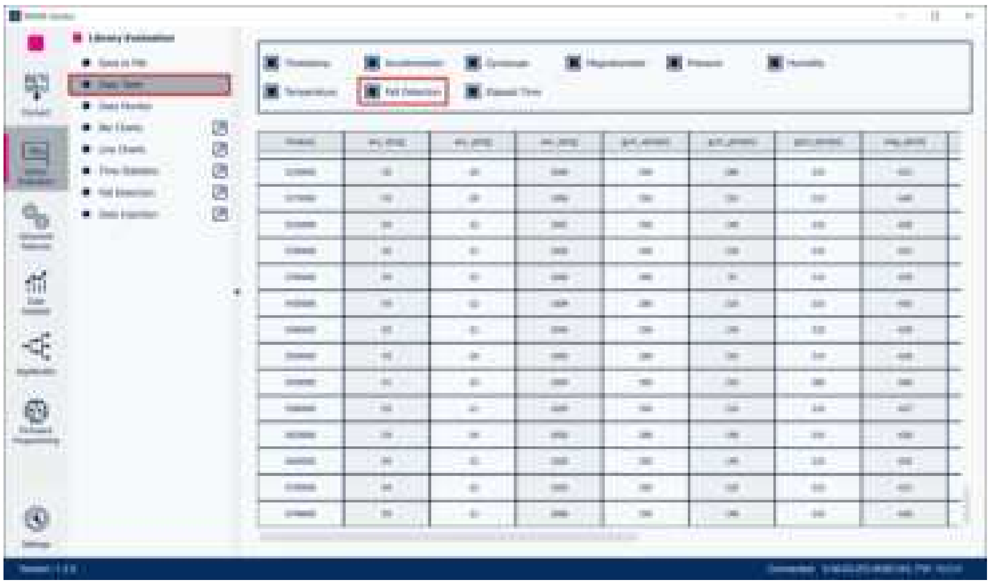


Step 3. When connected to a STM32 Nucleo board with supported firmware [**Library Evaluation**] tab is opened.

To start and stop data streaming, toggle the appropriate [**Start**] ▶ or [**Stop**] ■ button on the outer vertical tool bar.

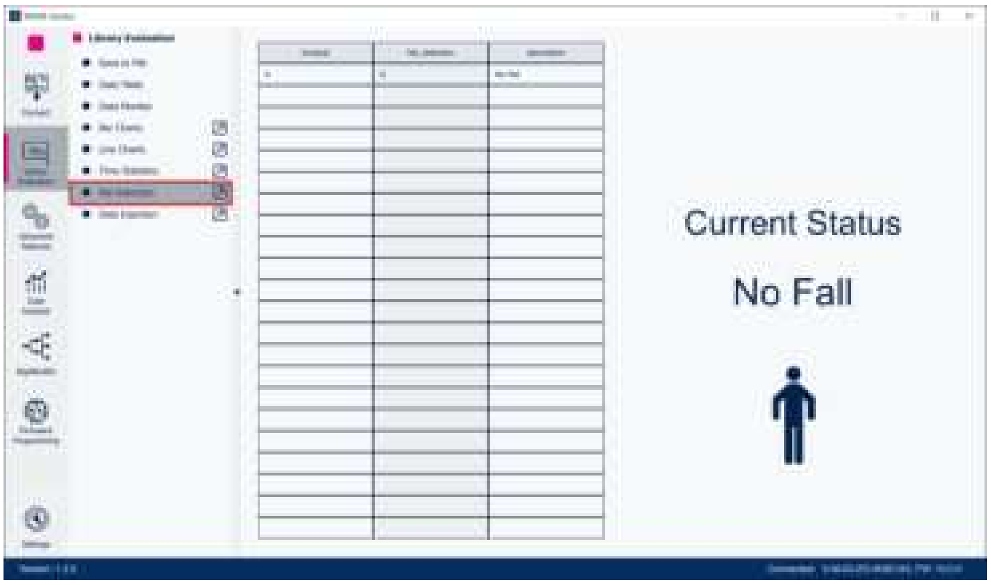
The data coming from the connected sensor can be viewed selecting the [**Data Table**] tab on the inner vertical tool bar.

Figure 4. **MEMS-Studio - Library Evaluation - Data Table**



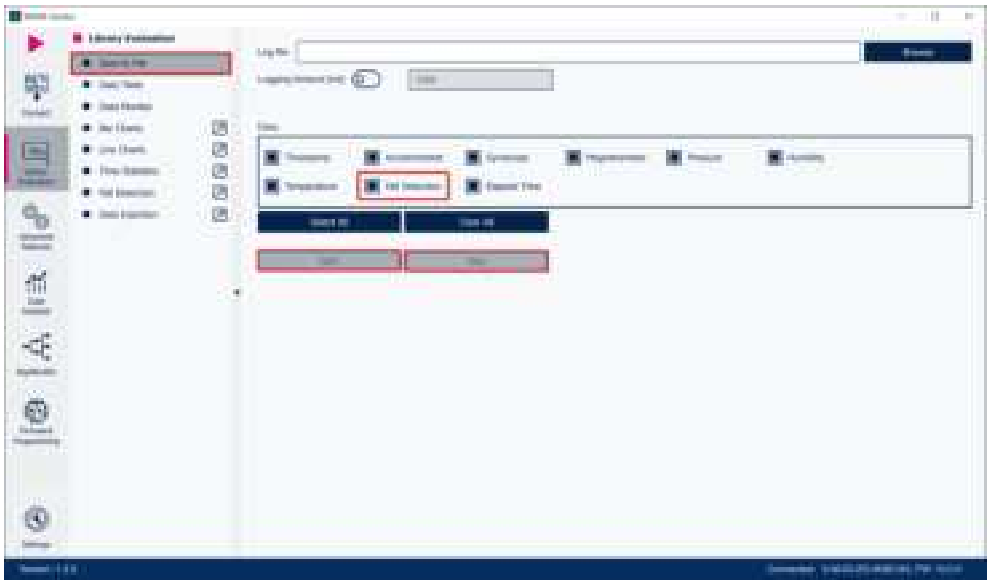
Step 4. Click on the [**Fall Detection**] to open the dedicated application window.

Figure 5. MEMS-Studio - Library Evaluation - Fall Detection



Step 5. Click on the [**Save To File**] to open the dataloging configuration window. Select the sensor and fall detection 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.

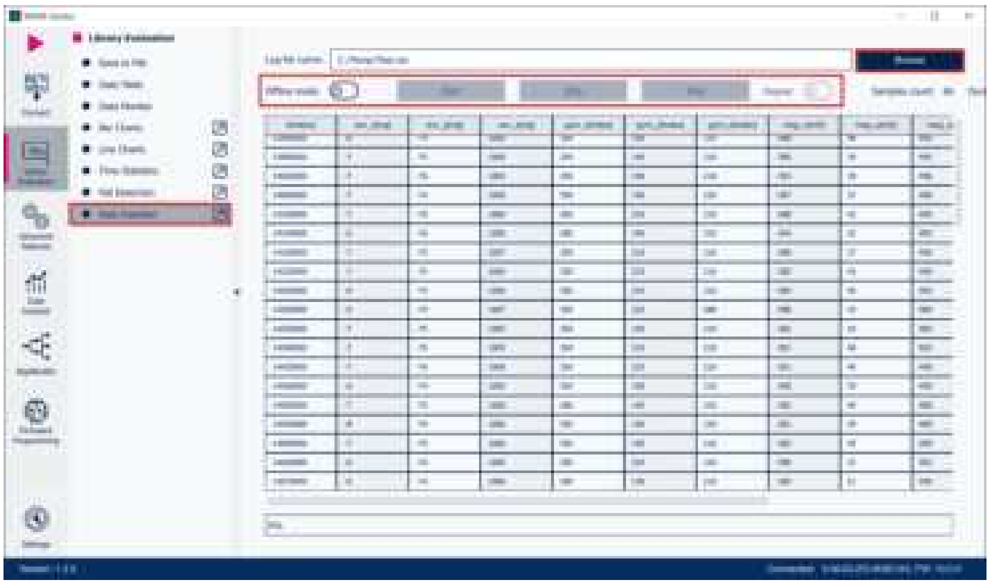


Figure 7. **MEMS-Studio - Library Evaluation - Data Injection**

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.

2.5 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
22-Sep-2017	1	Initial release.
06-Feb-2018	2	Added references to NUCLEO-L152RE development board and Table 2. Elapsed time (µs) algorithm.
21-Mar-2018	3	Updated Introduction and Section 2.1 MotionFD overview.
19-Feb-2019	4	Updated *Table 2. Elapsed time (µs) algorithm* and *Figure 2. STM32 Nucleo: LEDs, button, jumper*. Added X-NUCLEO-IKS01A3 expansion board compatibility information.
17-Sep-2024	5	Updated Section Introduction, Section 2.1: MotionFD overview, Section 2.2.1: MotionFD library description, Section 2.2.2: MotionFD 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