# Accelerometer Library - Documentation

## Library Description:

The Accelerometer library contains an assortment of functions to communicate and gain acceleration values from the LIS3DH accelerometer using a PIC24FJ64GA002 accelerometer. The library uses the I2C1 module of the microcontroller. Ensure that this module is not being used elsewhere. To use this library, connect the SDA1/SCL1 pins of the microcontroller to the SDA/SCL pins of the LIS3DH. Connect the SDO pin of the LIS3DH to ground, and the CS pin of the LIS3DH to Vdd. Connect both pins to a 10kΩ pull up resistor. Initialize the accelerometer with the initAccelerometer() function before using other functions.

## Hardware Description:

Microcontroller: PIC24FJ64GA002 | Microchip Technology
Adafruit LIS3DH Triple-Axis Accelerometer

## Full Documentation:

**void initAccelerometer()**

- Takes in no arguments, does not output anything
- Description: Initializes the accelerometer by initializing the I2C1 module of the microcontroller, and sending commands to initialize the LIS3DH.

**uint8_t accel_read(uint8_t address)**

- Arguments:
  - address - the register in the LIS3DH to read. Refer to Section 7 - Register Mapping (p. 31) of the LIS3DH datasheet manual for specific register addresses.
- Outputs: 8-bit value corresponding to the value read from the input address register of the LIS3DH

**void accel_write(uint8_t address, uint8_t data)**

- Arguments:
  - address - the register in the LIS3DH to write. Refer to Section 7 - Register Mapping (p. 31) of the LIS3DH datasheet manual for specific register addresses.
  - data - 8-bit value to write in the specified register address
- No outputs
- Description: Used to write a given register in the LIS3DH with the given data.

**int getXAcceleration()**

- No arguments
- Returns: x-axis acceleration measured by the sensor

**int getYAcceleration()**

- No arguments
- Returns: y-axis acceleration measured by the sensor

**int getZAcceleration();**

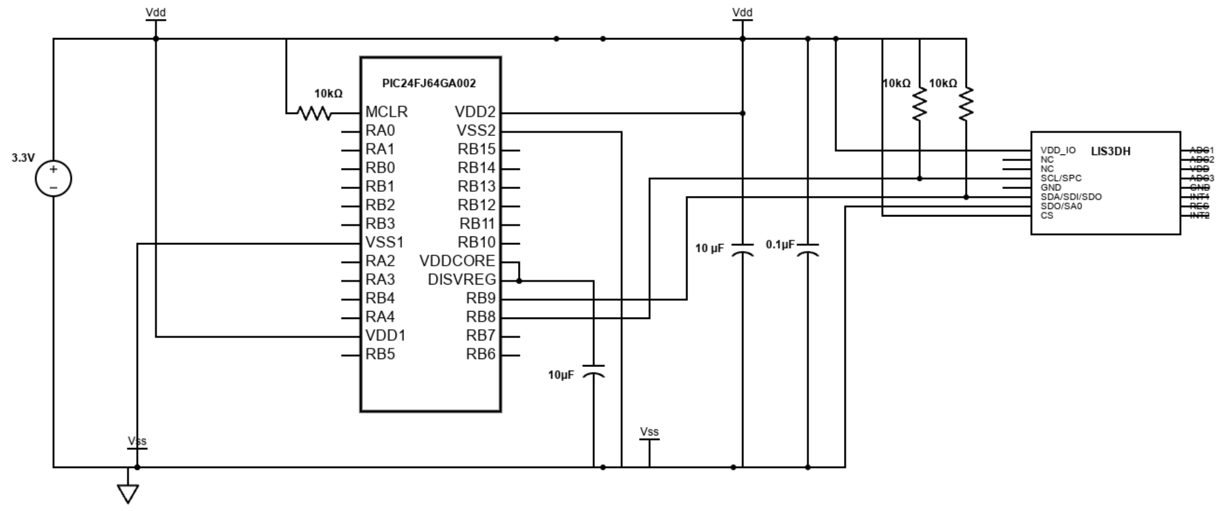
- No arguments
- Returns: z-axis acceleration measured by the sensor

**int movementDetected();**

- No arguments
- Returns: 1 if the accelerometer detected movement, otherwise return 0
- Description: The function will detect movement by seeing if the x, y, or z-accelerations

## Basic Usage Example:

### PIC24FJ64GA002 Circuit Schematic:



The LIS3DH accelerometer is connected to pins SDA1/SCL1 (RB9/RB8) on the PIC24FJ64GA002. 10kΩ pull up resistors must be connected between Vdd and SDA/SCL. Because SPI communication is not being used, pin SDO should be connected to Vdd and CSS to ground.

In the main code, include the “xc.h” library. Additionally, ensure to include the necessary flash configuration words into the main file (see Section 24.1 of the PIC24 Family Reference Manual for details).#include “Accelerometer.h” to use the Accelerometer library functions in the main code.

**Main Code Example:**

```
// CW1: FLASH CONFIGURATION WORD 1 (see PIC24 Family Reference Manual 24.1)
#pragma config ICS = PGx1          // Comm Channel Select (Emulator EMUC1/EMUD1 pins are shared with PGC1/PGD1)
#pragma config FWDTEN = OFF        // Watchdog Timer Enable (Watchdog Timer is disabled)
#pragma config GWRP = OFF          // General Code Segment Write Protect (Writes to program memory are allowed)
#pragma config GCP = OFF           // General Code Segment Code Protect (Code protection is disabled)
#pragma config JTAGEN = OFF        // JTAG Port Enable (JTAG port is disabled)


// CW2: FLASH CONFIGURATION WORD 2 (see PIC24 Family Reference Manual 24.1)
#pragma config I2C1SEL = PRI       // I2C1 Pin Location Select (Use default SCL1/SDA1 pins)
#pragma config IOL1WAY = OFF       // IOLOCK Protection (IOLOCK may be changed via unlocking seq)
#pragma config OSCIOFNC = ON       // Primary Oscillator I/O Function (CLKO/RC15 functions as I/O pin)
#pragma config FCKSM = CSECME      // Clock Switching and Monitor (Clock switching is enabled,
                                   // Fail-Safe Clock Monitor is enabled)
#pragma config FNOSC = FRCPLL      // Oscillator Select (Fast RC Oscillator with PLL module (FRCPLL))

Int main() {
        initAccelerometer(); // Initialize the accelerometer function before usage
// Get x, y and z acceleration values
        int x_accel = getXAcceleration();
        int y_accel = getYAcceleration();
        int z_accel = getZAcceleration();
}
```

## Advanced Usage Example:

The below main code shows a way in which an alarm and Neopixel can be used with the movementDetected() function to enable/disable a device which will sound an alarm when movement is detected. Uses three external libraries (Neopixel, Alarm & PushButton libraries).

```
// CW1: FLASH CONFIGURATION WORD 1 (see PIC24 Family Reference Manual 24.1)
#pragma config ICS = PGx1          // Comm Channel Select (Emulator EMUC1/EMUD1 pins are shared with PGC1/PGD1)
#pragma config FWDTEN = OFF        // Watchdog Timer Enable (Watchdog Timer is disabled)
#pragma config GWRP = OFF          // General Code Segment Write Protect (Writes to program memory are allowed)
#pragma config GCP = OFF           // General Code Segment Code Protect (Code protection is disabled)
#pragma config JTAGEN = OFF        // JTAG Port Enable (JTAG port is disabled)


// CW2: FLASH CONFIGURATION WORD 2 (see PIC24 Family Reference Manual 24.1)
#pragma config I2C1SEL = PRI       // I2C1 Pin Location Select (Use default SCL1/SDA1 pins)
#pragma config IOL1WAY = OFF       // IOLOCK Protection (IOLOCK may be changed via unlocking seq)
#pragma config OSCIOFNC = ON       // Primary Oscillator I/O Function (CLKO/RC15 functions as I/O pin)
#pragma config FCKSM = CSECME      // Clock Switching and Monitor (Clock switching is enabled,
                                   // Fail-Safe Clock Monitor is enabled)
#pragma config FNOSC = FRCPLL      // Oscillator Select (Fast RC Oscillator with PLL module (FRCPLL))

#include "xc.h"
#include "Accelerometer.h"
```

```
#include "PushButton.h"
#include "stdint.h"
#include "Alarm.h"
#include "Neopixel.h"
volatile int alarmOn = 0;


int main(void) {
   initAlarm();
   initNeopixel();
   initAccelerometer();
   initPushButton();

   loop();
   return 0;
}

void loop() {
   while(1) {
      if(isButtonPressed()) {
         blinkGreen();
         while(!isButtonPressed()) {
            if(movementDetected() && !alarmOn) {
               turnOnAlarm();
               alarmOn = 1;
            }
         }
         turnOffAlarm();
         alarmOn = 0;
         blinkRed();
      }
   }
}
```

Another advanced use case of the Accelerometer library is to customize the configuration of the LIS3DH by writing to the control registers. The initAccelerometer() function will write settings used by the library onto the control registers. After calling the initAccelerometer() function, the user can use the accel_write(uint8_t address, uint8_t data) function to write values to the control register. This can have the adverse effect of impacting the proper functionality of the library depending on how the settings are changed. Refer to the LIS3DH datasheet. The user can also use the accel_read(uint8_t address) function to read what is in a desired register. Refer to Section 8 - Register Descriptions of the LIS3DH datasheet for a description of the registers in the LIS3DH.

Example:

```
// CW1: FLASH CONFIGURATION WORD 1 (see PIC24 Family Reference Manual 24.1)
#pragma config ICS = PGx1          // Comm Channel Select (Emulator EMUC1/EMUD1 pins are shared with PGC1/PGD1)
#pragma config FWDTEN = OFF        // Watchdog Timer Enable (Watchdog Timer is disabled)
#pragma config GWRP = OFF          // General Code Segment Write Protect (Writes to program memory are allowed)
#pragma config GCP = OFF           // General Code Segment Code Protect (Code protection is disabled)
#pragma config JTAGEN = OFF        // JTAG Port Enable (JTAG port is disabled)


// CW2: FLASH CONFIGURATION WORD 2 (see PIC24 Family Reference Manual 24.1)
#pragma config I2C1SEL = PRI       // I2C1 Pin Location Select (Use default SCL1/SDA1 pins)
#pragma config IOL1WAY = OFF       // IOLOCK Protection (IOLOCK may be changed via unlocking seq)
#pragma config OSCIOFNC = ON       // Primary Oscillator I/O Function (CLKO/RC15 functions as I/O pin)
#pragma config FCKSM = CSECME      // Clock Switching and Monitor (Clock switching is enabled,
                                   // Fail-Safe Clock Monitor is enabled)
#pragma config FNOSC = FRCPLL      // Oscillator Select (Fast RC Oscillator with PLL module (FRCPLL))

#include "xc.h"
#include "Accelerometer.h"
#define CTRL_REG1 0x20
#define WHO_AM_I 0x0F

void setup();

void setup() {

        initAccelerometer(); // Initialize accelerometer before reading/writing manually

        // Using accel_read function to manually ensure that LIS3DH accelerometer
// returns proper address when reading the WHO_AM_I register
        if(accel_read(WHO_AM_I) != 0x33) {
                // Something is wrong!
        }
        accel_write(CTRL_REG1, 0b00001111); // Enable low power mode, ZYX axis on
// LIS3DH

}

Int main() {
        setup();
        while(1) {
                // main code logic …
        }
}
```