Experiment #3

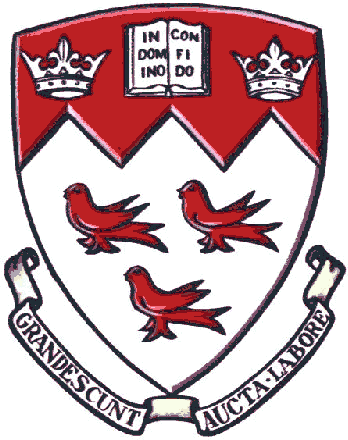
MEMS Accelerometer, Timers and Interrupts

Maxim Goukhshtein ID: 260429739

Olivier Laforest ID:260469066

Department of Electrical and Computer Engineering

McGill University, Montreal



March 16th, 2015

Group 3

ECSE 426 Microprocessor Systems

# Abstract

# Problem Statement

# Theory and Hypothesis

# Implementation

The implementation process can be divided into four parts, implementation of data acquisition of the MEMS accelerometer sensor, implementation of the keypad functionalities and implementation of the 7-segments display and overall operation of the program.

## MEMS Accelerometer Sensor

The model number of the Discovery board that is used is MB997C, therefore the MEMS accelerometer sensor model used is LIS3DSH. In order to acquire acceleration readings from the MEMS accelerometer sensor on the STM32F4 Discovery board, several components and interfaces need to be initialized and configured. The MEMS sensor on the Discovery board is an off chip sensor which means that the processor does not have direct access to it. The Serial Peripheral Interface (SPI) is used to retrieve the readings from the MEMS sensor and an External Interrupt/Event Controller (EXTI) is used to let the processor know that a measurement is ready. The initialization and configuration of the SPI, GPIO pins and the enabling of their associated clocks (i.e. APB2, ABH1) is done in a private sub function of the MEMS sensor driver's initialization function and there is no configuration options given to the designer that uses the driver. Consequently, the configuration these components will not be discussed here.

### MEMS Sensor Configuration

For the initialization and configuration of the LIS3DSH sensor, the following components of the initialization structure, *LIS3DSH\_initStruct*, had to be set up.

* *Power\_Mode\_Output\_DataRate*
* *Axes\_Enable*
* *Continous\_Update*
* *AA\_Filter\_BW*
* *Full\_Scale*
* *Self\_Test*

The requirement specification of experiment 3 required the reading of the MEMS sensor to be at least a 100 times per second, therefore, output data rate was set to a (i.e. *LIS3DSH\_DATARATE\_100*). To compute either the pitch or roll of the board, all three axes' acceleration are required, as a result, all three axes were enabled by ORing their respective enabling constant (e.g. *LIS3DSH\_X\_ENABLE*). The continuous update of the output registers was disabled (i.e. *LIS3DSH\_ContinousUpdate\_Disabled*) in order to ensure that the 16-bits output registers are not updated until both the lower and upper 8-bits registers have been read. This ensures that no values related to a different sample time are read [1]. The anti-aliasing filter bandwidth is usually set at half of the sampling frequency, it was therefore set to (i.e. *LIS3DSH\_AA\_BW\_50*). Computing the pitch and/or roll of the board only require the board to be subjected to gravity which corresponds to an acceleration of . As a result, the full scale value of the sensor was set to the smallest possible value of (i.e. *LIS3DSH\_FULLSCALE\_2*) in order to account for small acceleration while retaining the maximum possible sensitivity of . If it was known that the board may experience acceleration larger than , the full scale value should be increased at the cost of a lowered sensitivity. Finally, the self test feature was disabled (i.e. *LIS3DSH\_SELFTEST\_NORMAL*) since if it is left enabled, a biasing force is constantly applied to the sensor which is not desirable for normal operation of the designed system.

After the initialization of the MEMS sensor, the interrupts signaling that the data is ready to be read needed to be configured by initializing the *LIS3DSH\_DRYInterruptConfigTypeDef* parameters.

* *Dataready\_Interrupt*
* *Interrupt\_signal*
* *Interrupt\_type*

First, the interrupt signaling that data is ready needs to be enabled (i.e. *LIS3DSH\_DATA\_READY\_INTERRUPT\_ENABLE*). Second, the interrupt is set to be active high and pulsed (i.e. *LIS3DSH\_ACTIVE\_HIGH\_INTERRUPT\_SIGNAL* and *LIS3DSH\_INTERRUPT\_REQUEST\_PULSED*) as required by the specification of the experiment.

### External Interrupt Configuration

As mentioned earlier, the MEMS sensor is an off chip sensor and requires its interrupt to be channeled to the processor via an external interrupt. To achieve this, the following parameters of an EXTI initialization structure are configured (i.e. *EXTI\_InitTypeDef*).

* *EXTI\_Line*
* *EXTI\_Mode*
* *EXTI\_Trigger*
* *EXTI\_LineCmd*

The external interrupt line 0 (i.e. *EXTI\_Line0*) is selected since it is the line which is hardwired to the GPIO PE0 which is mapped to the interrupt signal of the MEMS sensor [2]. In order to use the EXT0, the high speed bus APB2 clock is enabled for SYSCFG and the EXT0 is connected to the GPIO pin PE0. The external interrupt mode was set to interrupt (i.e. *EXTI\_Mode\_Interrupt*). The mode could have been set to event as well, however, it was chosen to set it at interrupt. Rising edge triggering was selected (i.e. *EXTI\_Trigger\_Rising*) and the external interrupt was enabled.

### Nested Vector Interrupt Controller Setup

Finally, the interrupt request handler must be enable in the Nested Vector Interrupt Controller (NVIC) by configuring the following parameters of the initialization structure *NVIC\_InitTypeDef*.

* *NVIC\_IRQChannel*
* *NVIC\_IRQChannelPreemptionPriority*
* *NVIC\_IRQChannelSubpriority*
* *NVIC\_IRQChannelCmd*

The interrupt request channel is set to the one of EXTI0 (i.e. *EXTI0\_IRQn*). The preemption priority and sub priority are set to the highest priority and therefore the lowest number (i.e. 0) such that the processor handles interrupts generated by the MEMS sensor first. The channel is finally enabled to activate the handling of the MEMS interrupts. The MEMS sensor is now fully configured and ready to use.

### Data Acquisition and Conversion

The specifications of experiment 3 required that either the pitch or the roll of the board be displayed in real time. The pitch was selected to be displayed. With the MEMS sensor initialized and configured, it is now generating acceleration measurements 100 times per second. Every time a measurement is ready, an interrupt is generated, the processor switch to the corresponding handler routine which triggers the retrieval of the acceleration readings of the MEMS sensor through its driver API call to compute the associated pitch angle. The raw acceleration readings are normalized and adjusted by multiplying them with the calibration matrix explained in the theory section. These normalized X, Y and Z acceleration values are then converted into pitch angle by using ( ).

### Filtering

Given the fact that the pitch angle values are computed from noisy unfiltered raw accelerations which exhibit sharp fluctuations, the computed values also contain noise, and therefore must be filtered. In order to remove the noise from the pitch angle values, a 1D Kalman filter is used. The selection of the initial Kalman filter parameter was done experimentally.

## Keypad Interface

In order to allow the user to input a desired board pitch angle, a 4X4 keypad was used. The GPIO pins PE7 to PE14 were used to connect and allow the program to interact with the keypad. These GPIOs need to be initialized and configured. The timer TIM3 was used to implement a debouncing algorithm for the keypad, however, its initialization and configuration is discussed later in this section as TIM3 was also used in the implementation of the 7-segments display.

### GPIO Configuration

The be able to use the GPIOE ports the clock on the AHB1 bus needs to be enabled, however, it already has been enabled by the initialization of the MEMS sensor private driver function. The configuration of the GPIOE pins 7 to 14 is done by setting the following parameters of the *GPIO\_InitTypeDef* structure.

* *GPIO\_Pin*
* *GPIO\_Speed*
* *GPIO\_Mode*
* *GPIO\_OType*
* *GPIO\_PuPd*

As mentioned above, the port PE7 to PE14 are used to connect the keypad, however, the ports PE7 to PE10 are connected to the rows of the keypad and are configured as outputs while ports PE11 to PE14 are connected to the columns of the keypad and are configured as inputs. The speed of the port was set to (i.e. *GPIO\_Speed\_50MHz*) which is adequate for the operation of the keypad. Finally, the output type of the ports were set to push/pull (i.e. *GPIO\_OType\_PP*) since the ports need to be driven high or low and the default push/pull configuration of the port was set to pull the ports up (i.e. *GPIO\_PuPd\_UP*) since the algorithm to detect a key stroke was design as active low (i.e. a pressed key represent a logical 0 and a key which is not pressed represent a logical 1).

### Keypad Operation

The keypad is composed of 4 columns and 4 rows. When a key is pressed, it electrically connects the corresponding row and column. Therefore, to identify which key is pressed by the user, an algorithm was designed such that it subsequently sets one of the row active, and while the row is active, it scans the columns to see which one is active (if the user is currently pressing a key). By knowing the active row and column, it can be easily found which key is pressed by a simple mapping of row/column pattern to each key since each row/column pattern can only be associated with only one key. As long as the user has not successfully entered the desired angle and pressed the enter key, the scanning of the row/column pattern runs in a while loop. The frequency at which this while loop runs is equal to the frequency of the TIM3 timer which is . Given the frequency of the scanning loop and the bouncing effect of the mechanical switch of the keys of the keypad, an algorithm that ensured that a key pressed by the user is only counted once was implemented. This algorithm consists in deactivating the monitoring of the keypad for a fixed amount of TIM3 clock cycle as soon as a key is detected as pressed, which effectively reduces the frequency at which the keypad is scanned. After experimental tuning, is was found that a scanning frequency of was adequate for the proper operation of the keypad. The '#' key was selected on the keypad to implement the enter function. Several safety features which ensure that the angle entered by the user is in the allowed range of to and that only numerical keys or the enter key are valid inputs were added. Finally, if the user has entered an angle between and , the program automatically proceeds without requiring the user to press enter as it is not necessary.

## 7-Segments Display

A 4 digit 7-segments display is used to provide visual feedback to the user. Only one digit of the 7-segments display can be displayed at once, therefore, a timer is needed to cycle and display each digit sequentially. The hardware timer TIM3 was used to provide the necessary delay. GPIO pins PD0 to PD4 and PD6 to PD12 were used to connect and operate the display.

### TIM3 Configuration

Initialization of the TIM3 timer involves the activation of the APB1 bus clock. To configure the TIM3 timer, the following parameters need to be set.

* *TIM\_Prescaler*
* *TIM\_CounterMode*
* *TIM\_Period*
* *TIM\_ClockDivision*

According to [2], the base frequency of the TIM3 timer is . In order to achieve the desired digit cycle frequency of , which was found adequate by experimental tuning, the prescaler was set to its maximum possible value of , which yields an intermediate frequency of . The mode in which the counter of the TIM3 timer counts was set to counting up (i.e. *TIM\_CounterMode\_Up*). The period of the timer was set to 8 in order to achieve the operating frequency of . The base clock division was set to 1 since no division of the base clock was required.

### NVIC Configuration

Once the TIM3 is initialized, the interrupt request corresponding to the TIM3 timer in the NVIC needs to be initialized. The TIM3\_IRQ is initialized in the exact same way as the EXTI0\_IRQ described in the corresponding section, except that the preemption priority of the TIM3 is set to 1. In other words, the priority of TIM3 Interrupt request is lower than the one of the MEMS sensor's interrupt. This is done because reading the MEMS sensor's measurement has a higher priority than displaying the next digit of the 7-segments display.

### GPIO Configuration

The GPIO initialization parameters are the same as one presented in the keypad description in this section. The parameters of the GPIO used for the 7-segments display are initialized to the same value as the ones used to operate the keypad, except that all of the GPIOs for the display are configured as output and that no pull is applied to the ports. GPIO pins PD0 to PD4 and PD6 to PD7 are connected to the pins representing segment A to G of the 7-segments display respectively. GPIO PD8 is connected to the decimal point pin and PD12 to the degree sign pin. GPIO pins PD9 to PD11 are connected to the base of three separate NPN transistors which act as switches to close the circuit to ground of each digit of the 7-segments display. The port PD9 closes the circuit of the first digit on the display, PD10 the second digit and PD11 the third digit of the display.

### 7-Segments Display Operation

Three modes of operation are required for the 7-segments display. In the first mode, the board's pitch angle is smaller than the user's desired angle by more than 5 degrees and the display outputs the word "UP" signifying that the user should move the tip of the positive X axis of the LIS3DSH MEMS sensor as shown in figure 1 up. In order to display a letter or a number, the necessary segment's port which realizes the desired letter (or number) are set high before the digit of the display is lit.

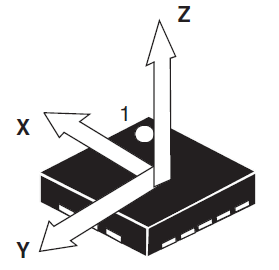


Figure 1: Positive axis orientation of the LIS3DSH MEMS Accelerometer sensor [1].

In the second mode of operation of the display, the pitch angle of the board is larger than the desired angle by more than 5 degrees and the display outputs the letters "dn", which stands for "down", signifying the user to move tilt the tip of the positive X axis down. In the final mode of operation, the board's pitch angle is within of the desired angle and the display outputs the current board angle in real time. To correctly display the pitch angle in real time, the program identifies the magnitude of the board's pitch angle to determine if and where the decimal point should be lit, and then parses the correct digit of the pitch angle to be displayed. The three most significant digits of the pitch angle are displayed one at a time and are subsequently displayed in a cyclic manner based on the clock cycle of the TIM3 timer.

## Overall Operation

The overall operation of the program is separated in two sections. First, the program waits for the user to input his desired target pitch angle. Once the desired angle is correctly entered, the program enters an infinite loop which performs two tasks in its body. When a measurement of acceleration done by the MEMS sensor is ready, the program retrieves the measurement and updates its pitch angle. Otherwise, when the TIM3 timer clock cycle expires, the program proceeds to display the next digit of the current mode of operation on the 7-segments display.

# Testing and Observations

# Conclusion

In the course of this experiment, using the off chip LIS3DSH MEMS accelerometer sensor of the STM32F4 Discovery board, a 4X4 keypad and a 4 digits 7-segments display, a system which prompts the user to input a desired pitch angle for the Discovery board and outputs a visual feedback to guide the user was implemented. Measurements of the gravitational acceleration experienced by the board combined with the use of a 1D Kalman filter to reduce the noise present in the raw data were used to compute the real time pitch angle of the board. The Kalman filter's initial parameter values were determined experimentally. The MEMS sensor communication with the microprocessor of the Discovery board was achieved through the use External Interrupt/Event Controller and Serial Peripheral interface. The keypad and the 7-segments display were connected to the board through GPIO and their correct operation was synchronized by the hardware timer TIM3. All of these elements were combined to provide the user with a visual feedback which indicates the correct direction in which the board should be tilted to achieve the desired pitch angle. If the board is within 5 degrees of the desired angle than the current real time pitch angle is displayed.

# References

|  |  |
| --- | --- |
| [1] | "LIS3DSH MEMS digital output motion sensor ultra low-power high performance three-axis “nano” accelerometer," October 2011. [Online]. Available: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00040962.pdf. [Accessed 16 March 2015]. |
| [2] | "RM0090 Reference manual STM32F405xx, STM32F407xx, STM32F415xx, and STM32F417xx advanced ARM-based 32-bit MCUs," STMicroelectronics, 2014. [Online]. Available: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/reference\_manual/DM00031020.pdf. [Accessed 21 February 2015]. |
| [3] | R. Faragher, "Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation," *IEEE Signal Processing Magazine,* pp. 128-132, 20 August 2012. |
| [4] | A. Suyyagh, "ECSE 426 Microprocessor Systems Lab 1: One-Dimensional Kalman Filter," 2015. |
| [5] | "UM1472 User manual Discovery kit for STM32F407/417 lines," STMicroelectronics, 2014. [Online]. Available: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user\_manual/DM00039084.pdf?s\_searchtype=keyword. [Accessed 21 February 2015]. |
| [6] | "STM32F405xx/STM32F407xx Datasheet," STMicroelectronics, 2013. [Online]. Available: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00037051.pdf. [Accessed 21 February 2015]. |
| [7] | A. Suyyagh, "ECSE 426 Microprocessor Systems Lab 3: MEMS Accelerometer, Timers and Interrupts," 2015. |

# Appendix

Matlab code used for testing the Kalman filter parameters:

% 1D Kalman filtering of data using with initial Kalman state parameters

% p, r and q.

function [filtered] = KalmanFilter1D(data, p, ~, r, q)

filtered = zeros(length(data), 1);

% filtering

x = data(1);

for i = 1:length(filtered)

p = p + q;

k = p / (p + r);

x = x + k \* (data(i) - x);

p = (1 - k) \* p;

filtered(i) = x;

end

% plotting result

plot(1:length(filtered), data, 1:length(filtered), filtered, 'r');

title(['Raw sensor data and Kalman filtered data (r = ', num2str(r),...

', q = ', num2str(q), ')']);

end