Lab 2 Report

ECSE 426 – Microprocessor Systems

Group 7

Gregoire Martin – 260364773

Patrick White – 260353491

October 21st 2013

**Abstract**

The primary goals of this experiment are to provide a simple graphical output of the processor core’s temperature using the STM32Fx board’s LEDs, to implement a simple pulse width modulation (PWM) algorithm and demonstrate its correctness using the board’s LEDs, and to provide to the user a method of selecting between the modes of operation using a button. This experiment involved the use of sensors, timers, basic input and output, as well as some basic signal processing for the sensors and for a button.   
\*\*\*\* NEED TO ADD DATA AND CONCLUSIONS \*\*\*\*

**Problem Statement**

The end goal of the experiment is to have a simple LED display indicate whether the board is heating up, cooling down, or maintaining its temperature, as well as demonstrating PWM using the same LEDs and allowing the user to select the mode of operation. The problem can effectively be broken down into five parts:

* Acquiring data from the temperature sensor
  + The data from the temperature sensor must be acquired as a voltage value and converted into a temperature using a formula provided in the datasheet.
  + The data must be sampled at a high enough rate to be useful.
  + The raw voltage readings are provided in analog format, thus the data must be converted to digital format in order for the processor to be able to use it.
* Filtering noise out of the signal
  + The signal is expected to be very noisy. A filter must be used to improve the quality of the signal and get rid of the unwanted noise.
* Updating the LEDs according to the temperature
  + The four main LEDs on the board must light up in a clockwise fashion with only one LED on at a time when the core temperature is increasing, and they must turn on in a counter clockwise fashion with only one LED on at a time. The LEDs must change whenever the temperature changes by 2 degrees Celsius.
* Developing the PWM algorithm
  + Describe what PWM is here
* Providing the user a way of selecting between the two modes of operation
  + The user should be able to select between PWM and temperature tracking by simply pressing a button on the board.
  + The signal from the button will be noisy because of contact bounce, and a way must be devised to obtain a clean reading from the button

These five aspects will allow the board to provide a simple display to the user that describes the temperature trend of the processor’s core, as well as demonstrating PWM on the board’s LEDs and allowing the user to easily select the desired mode of operation.

**Theory and Hypothesis**

The values obtained from the temperature sensor will originally be voltages in an analog format. Because the readings are in an analog format, Analog-Digital conversion must be used to convert the data into a format that the processor will be able to use.

From the STM32Fx datasheet (add reference), the formula to convert the voltage reading from the temperature sensor into a temperature in degrees Celsius is

Where V25 is the voltage measured at 25 degrees Celsius, or 0.76 V.

It is expected that the voltage readings from the sensor will have unwanted fluctuations which will make appear that the temperature is varying far more than it actually is. This noise can be caused by many sources; electromagnetic interference, thermal noise, quantization noise from the ADC among other things. The signal must be processed in such a way to filter as much of this noise as possible. (talk about moving average filter)

Provide PWM theory and hypothesis for window (add figure probably to demonstrate pwm)

In order to obtain a clean reading from the push button, some basic signal processing will have to be applied to the reading because of contact bounce. (add reference and probably a figure to demonstrate the bouncy signal) Contact bounce occurs because many buttons are made of springy metals. When the spring comes into contact with the electrical contacts, it will result in a “bouncy” signal where the bit value may rapidly pulse between 0 and 1 for a short period of time, preventing the programmer from reliably knowing if the button was pressed. It will be necessary to “debounce” the button using some very basic software functions. A simple way of debouncing the button is to check for both a press and a release. If the bit indicating a press is set, wait a short period of time, and then check if it’s been unset. This gives the signal time to be fully asserted, and fully de-asserted before the bit readings are taken.

**Implementation**

**Testing and Observations**

**Conclusion**