

# EENG460a/CPSC436/ENAS960

## Programming Assignment #1

Professor Andreas Savvides  
*andreas.savvides@yale.edu*

Teaching Assistant: Dimitrios Lymberopoulos  
*dimitrios.lymberopoulos@yale.edu*

September 21, 2004

Class url: <http://www.eng.yale.edu/enalab/courses/eeng460a/>  
**Due date: October 6, 2004.** Demonstration during TA's office hours.

### 1 Introduction

The purpose of this assignment is to familiarize you with the core concepts of embedded systems programming. For this task you will use the XYZ sensor node. After finishing this assignment you should be fluent with the following:

- The XYZ's architecture.
- Configuring the GPIO pins of the OKIML67Q5002 processor.
- Programming and using the OKIML67Q5002's hardware timers.
- Sampling the on-board sensors.
- Programming and using the UART interface

The assignment consists of two parts that are described in detail in the next sections. The starting code for this assignment is posted on the class website.

### 2 Part 1 (GPIO and Timers)

In this part you have to create a 3-bit binary counter on the XYZ sensor node. The value of the counter should be increased by 1 every second. When the counter overflows (current value of the counter is 7 and a second has elapsed) it should be automatically reset to its initial value: 0. The counter continues to count seconds even after an overflow. The current value of the counter should be monitored at all times on the 3 on-board LEDs (red, yellow and green). The most significant bit (MSB) of the counter should be monitored on the red LED, while the least significant bit (LSB) of the counter should be monitored on the yellow LED.

A high level flowchart of the binary counter application can be seen in Figure 1. First, you have to select and configure the GPIO pins of the processor that are necessary to the application. Then, you have to configure one of the available hardware timers appropriately, so that you can count seconds. After initializing the hardware timer, create a timer handler that implements the 3-bit binary counter.

### 3 Part 2 (Sensor sampling, ADC, UART)

In this part of the assignment, you will learn how to use the internal ADC of the OKI processor to sample two of the on-board sensors: the light sensor and the accelerometer. Your task is to periodically sample these sensors and output the sample values either on the on-board LEDs or at the UART port.

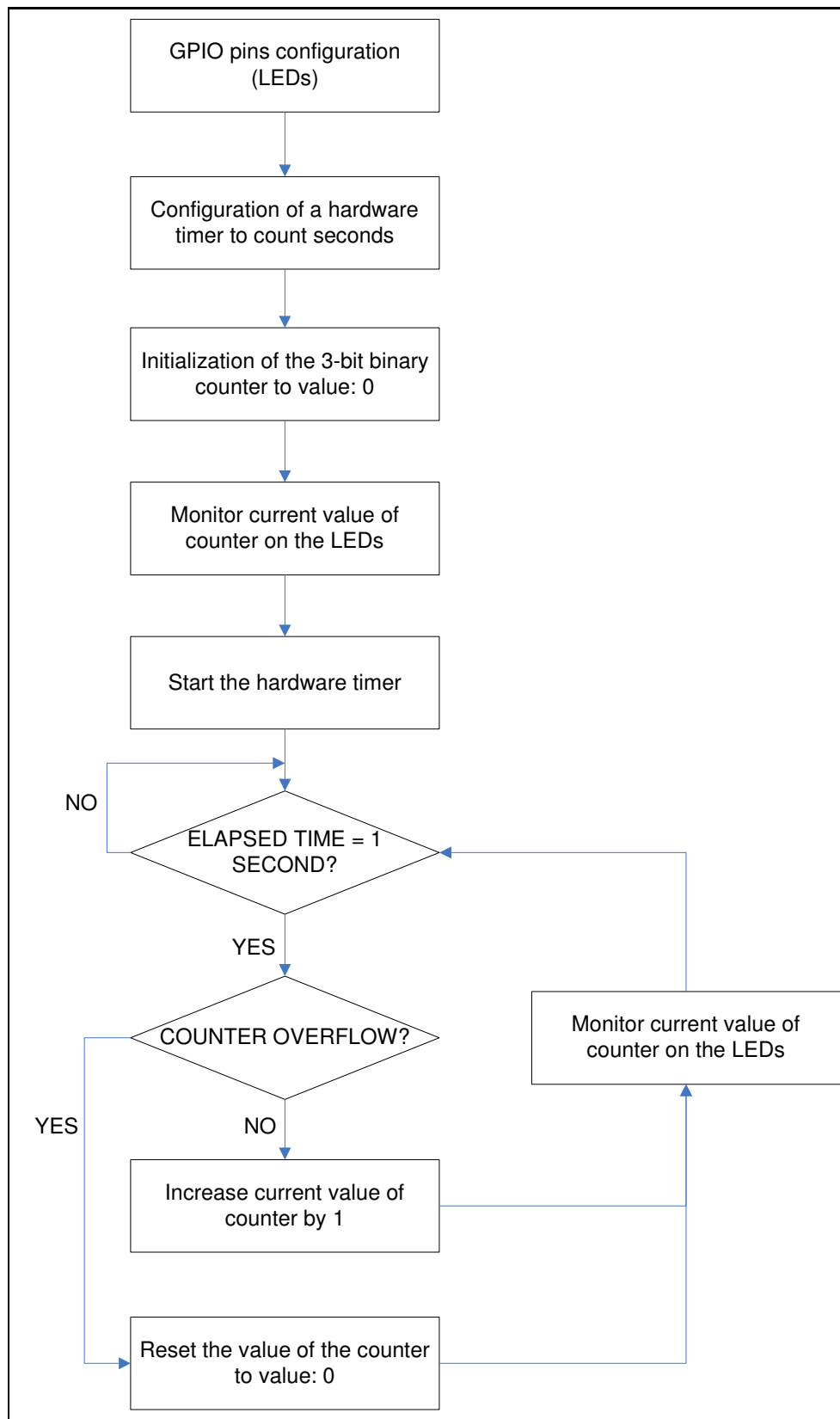


Figure 1: High level flowchart of the 3-bit binary counter application.

### 3.1 Light sensor sampling

In this section you have to sample the on-board light sensor every 5 seconds. Based on the value sampled each time, your program should output a specific pattern on the on-board LEDs. All the possible patterns are described below:

- if the light sensor voltage is less than 1V only the yellow LED should be turned on.
- if the light sensor voltage is more than 1V and less than 2V then the yellow and green LEDs should be turned on.
- if the light sensor voltage is more than 3V then all the LEDs should be turned on.

Initially all the LEDs are turned off. You are free to use the configuration of the ADC that is more convenient to you.

### 3.2 Accelerometer sampling and UART

In this part you have to use the code that was developed in the previous subsection. Your task is to expand this code by implementing the sampling of the on-board accelerometer. You have to sample the on-board accelerometer every 5 seconds. The light sensor data should still be monitored on the on-board LEDs as it is specified in the previous subsection. In terms of this part of the assignment, the sampled values of both the accelerometer and the light sensor should be output at the UART port of the XYZ sensor node. Therefore, you must first configure and program appropriately the processor's UART registers. In order to be able to output data to the UART port you will have to implement a UART driver.

## 4 Deliverables

The deliverables for this assignment are the following:

1. A demonstration of your implementation to the TA during his lab hours. During the demonstration you might be asked to explain in detail your implementation.
2. A zip file containing the following files:
  - A zip file containing ALL the files for the first part, including makefiles and executables.
  - A zip file containing ALL the files for the second part, including makefiles and executables.

If for any reason you changed the way you are compiling your code or you want to make specific comments about your implementation, please include a README file in the zip files described above. Your files (a single zip file) should be sent to: *dimitrios.lymberopoulos@yale.edu* with subject: **PA#1**. The name of the zip file should follow the following format: **lastname\_PA1**.

## 5 Getting help and answers to your questions

If you need help or you have questions about the assignment you can contact the teaching assistant of the course in the following ways:

- Email: *dimitrios.lymberopoulos@yale.edu*
- Stop by the lab during the lab hours:
  - Wednesday 2:00pm - 4:00pm
  - Tuesday 4:00pm - 6:00pm