**Lab 6 Sensing the Light**

**Student Name: Joseph Tomalewski**

**Date: 12/6/22**

**Purpose of Lab**

The purpose of the lab is to program a timer using either the polling or interrupt method to implement into the circuit we create on a breadboard using the Arduino. Using the Arduino, we will be utilizing the AVR architecture and instruction set to main program everything in assembly with the setup programs in C. This lab is by far the most in depth we go into, and by memorizing the registers that we must program is simply too hard so we will use a provided data sheet.

**Description of Solution(s)**

First, before programming a polling or interrupt timer, I would first have to program in assembly the initialization of the sensor, along with converting the sensor input to a digital format for the computer to read properly. Setting DIDR0 and ADCSRA to enable ADC and getting it prepared.

Next, I start programming the conversion block which is function readSensor (). In this function I set the MUX, setting the MUX will set the analog input pins that will be used and will change the result of the conversion to be left or right adjusted within the register. After setting the ADCSRA with the conversion bits, it will now check if bit 6 is cleared in the ADCSRA to continuously loop until the conversion is stopped and over, to then load the result to register R24 from the result data register ADCH.

Finally, I would start programming a timer to pick up the signal from the light sensor and print it on the serial monitor every exact second. To program the exact second, I decided to use the Timer1 built onto the AVR. Using this Timer1 will grant me a 16-bit register to use for overflowing the timer. Since I decided to use normal mode, I will use prescaler 1:256 which will be just enough for me to overflow using 62,500 cycles that will equate to exactly 1 second.

I also use the same configuration for my interrupt timer that I also implemented. However, the interrupt timer I implemented is programmed in C, but it is still also used in conjunction with my assembly code to initialize and read the light sensor.

**Test Results**

I would first test the initialization and read sensor programs that I coded in assembly by noticing if the output is correct. Since I am programming a light sensor, it should be outputting values between 0x0 and 0xFF. 0xFF being the brightest that is being detected. So, using this information I would test if it were programmed properly to see if it is outputting corresponding values depending on how much light is on the sensor and I can manipulate this with covering the sensor with my hand. At first, I had an issue with using the wrong register initializing the conversion, which I got fixed immediately after talking to a TA.

Lastly, I would program the delay correctly by using the informative class book about the AVR microcontroller. It has tables showing prescaler configurations that match up to exactly 1 second when configuring the timer correctly. I would use one of these configurations and program them accordingly, which seems accurate as the light sensor values are being outputted every second.

**Discussion**

I really enjoyed doing this lab and I thought this was a moderately easy lab to complete due to the materials provided, and a very informative video that was very clear on how to proceed with programming the main sections needed for the light sensor to work. I learned a lot as expected of spending as much time as I did read what each control register does and just organizing and programming in assembly is fun and can really check your understanding on how each register affects everything else.

In my opinion, I think it would perhaps be better instead of only programming the light sensor, maybe if we were tasked with programming something to work with the light sensor or just something extra so we could better understand how to setup the ADC and other control registers. For example, maybe programming the data received from the light sensor to send to a 7-segment LED.

**Contribution to Teamwork**

This was an individual lab, all work produced by Joseph Tomalewski.

**References**

AVR Instruction Set: <http://ww1.microchip.com/downloads/en/devicedoc/atmel-0856-avr-instruction-set-manual.pdf>

AVR Data Sheet:

