EEET2490 – Embedded Systems: Operating Systems and Interfacing

LABORATORY 1

**ATmega32 Microcontroller Essentials & Functions**

**Lecturer:** Dr Andrew Smith

**Student Name:** Tran Thi Hong Phuong

**Student ID:** s3623386

**Date:** 10th August, 2018

**Table of Contents**

[1. INTRODUCTION 3](#_Toc519623822)

[2. TASK 1 – Toggling LEDs 3](#_Toc519623823)

[3. TASK 2 – Controlling the Brightness of LEDs 3](#_Toc519623824)

[4. TASK 3 – Controlling LEDs with a Push Button 3](#_Toc519623825)

[5. TASK 4 – Controlling the Brightness of an LED with a Timer 3](#_Toc519623826)

[6. TASK 5 – Toggling an LED Using a Timer 3](#_Toc519623827)

[7. TASK 6 – Toggling the Flashing Rate of an LED Using Interrupts 3](#_Toc519623828)

[8. CONCLUSION 3](#_Toc519623829)

[9. REFERENCES 3](#_Toc519623830)

[10. APPENDIX 3](#_Toc519623831)

# INTRODUCTION

# TASK 1 – Toggling LEDs

## Sourcing

In this task, the OUSB is used to toggle an LED via a General Purpose Input/Ouput (GPIO) pin. For each GPIO pin of the ATmega32 microprocessor, the output DC voltage and current is 5V and 40.0mA respectively [1]. However, common LEDs operates ideally at forward voltage varying between 1.7 to 3.5V and forward current at around 20mA [2]. The LED used in this task is a red one, thus the typical operating voltage is 2.2V [2]. Therefore, some resistance should be added to the circuit in serial with the LED, so as to limit the current and ensure the life of the LED. The resistance value is calculated as followed:

Since a 140 was not available, it was replaced with a 150 resistor instead. Below is the schematic of the circuit, where the microcontroller is sourcing the LED.

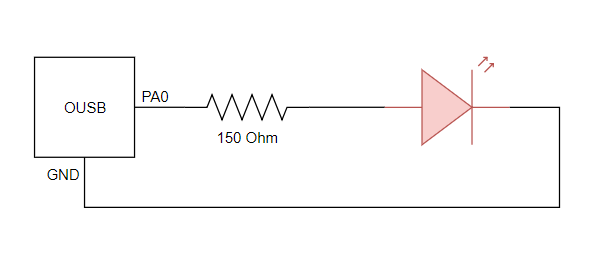


Figure 1. OUSB sourcing LED

Refer *Appendix 1* for the full code of the task. In the first scenarior, the program is designed to toggle the LED every second, which means that the Pin 0 of Port A (PA0) is set to output HIGH signal, followed by a delay of 1000 millisecond, and then set to output LOW signal with another delay of the same time. The same method is applied to the second scenarior where the LED is required to toggle five times every second, or five times every 1000 millisecond, so the delay between each command is reduced to 200 millisecond. The waveform result of this program is presented below.

## Sinking

Next, a green LED is used to create a connection where the OUSB board is sinking it, while the red LED circuit is kept as above. The HIGH voltage source is now set to VCC instead of a GPIO pin, and GND is substituted with GPIO Pin 1 of Port A (PA1). The schematic is described as followed:

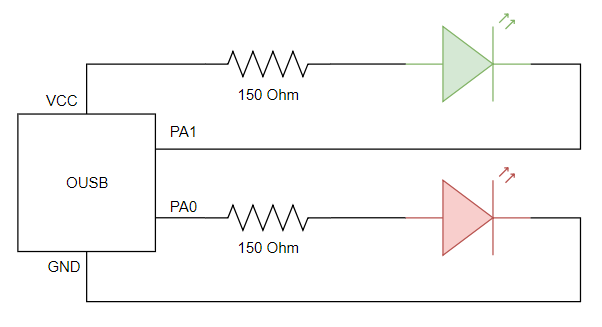


Figure 2. OUSB sourcing red LED and sinking green LED

In sourcing connection, the LED takes HIGH voltage source from the GPIO pin, for this case is PA0. When PA0 is at logic 1, it outputs 5V signal thus creating current flow through the LED, turning it on. When PA0 is at logic 0, there is no current flow, and LED is off. On the other hand in sinking connection, when PA1 is at logic 1, there is no voltage drop thus no current flow through the LED, hence it is off. When PA1 is at logic 0, there is a voltage difference between two ends of the circuit creating current flow through the LED, thus it is on. In another word, the behaviour of a sunken LED contrasts with a sourced one.

In the code, within the infinite main loop, PA0 and PA1 are set to logic 1 for 200ms at the same time, then to logic 0 for another 200ms. When the code is executed, the two LEDs behave differently, they blink in sequence. When red LED is on, green LED is off, and vice versa. The waveform result of this program is presented below.

# TASK 2 – Controlling the Brightness of LEDs

The activity of this task is learning how to control the brightness of an LED using fundamental principles, without the help of special functions provided by the microcontroller. The electronic circuit for this task is the same as that described in *Figure 1* in *Task 1* section. Adjusting brightness of an LED can be implemented using Pulse Width Modulation, or PWM. PWM is a technique to retrieve analog results from digital on-off signals. This on-off pattern can simulate voltage level varying between full on (for the OUSB board, its output voltage is 5V) and off (0V) by changing the duty cycle, which is the amount of time the signal is at HIGH state over the total time of one cycle [3].

## Toggle LED at the rate of 100Hz

The LED is required to toggle at the rate of 100Hz, so the time period of one on/off cycle is:

So the LED is toggled every . The waveform result of this program is presented below.

## Dim the LED using different duty cycles

The second activity of this task is modifying the above to adjust the brightness of the LED. In order to create a visual illusion that the LED is steadily dimmer than its actual brightness, the flashing, toggling on and off of the LED should not be visible to human eye. This refers to a concept called flicker fusion threshold, a frequency high enough to create persistence of vision, which is when a light stimulus appears steady to the average human observer [5]. The flicker artifacts are reported to be relatively invisible to most viewers at the rate above 30Hz [6]. Therefore, 100Hz toggle rate setting in *Section 3.1* above is more than enough to achieve a seemingly stable light output from the LED.

The next step is to apply PWM technique to adjust the brightness of the LED. The brightness of an LED is proportional to the amount of time it stays at HIGH state within each cycle. Without PWM configuration, for one cycle, the amount of time the LED stays at HIGH state is equal to that at LOW state, which is 50% duty cycle. This duty cycle can be modulated by changing the delay time between each LED toggle. If this duty cycle is increased, which means the toggle on delay is longer than the toggle off delay, the LED will appears brighter, and vice versa. To justify this statement, the code provided in *Appendix …* is designed to drive the LED at different duty cycle values (10%, 20%, 30%, 40%, 50%, 75% and 100%) and observe the brightness difference in each case. The waveform result of this program is presented below.

# TASK 3 – Controlling LEDs with a Push Button

# TASK 4 – Controlling the Brightness of an LED with a Timer

# TASK 5 – Toggling an LED Using a Timer

# TASK 6 – Toggling the Flashing Rate of an LED Using Interrupts

# CONCLUSION

# REFERENCES

# References

|  |  |
| --- | --- |
| [1] | "ATmega32A Datasheet Complete," Atmel Corporation, Aug 2016. [Online]. |
| [2] | "Dangerous Prototypes," 11 Oct 2012. [Online]. Available: http://dangerousprototypes.com/docs/Basic\_Light\_Emitting\_Diode\_guide. [Accessed 20 July 2018]. |
| [3] | "What is a Pulse Width Modulation (PWM) Signal and What is it Used For?," National Instruments, May 2018. [Online]. Available: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019OkFSAU. [Accessed 23 Jul 2018]. |
| [4] | "Period And Frequency," Siyavula, [Online]. Available: https://www.siyavula.com/read/science/grade-10/transverse-waves/08-transverse-waves-06. [Accessed 30 Jul 2018]. |
| [5] | S. Davis, "Auditory and visual flicker-fusion as measures of fatigue," *The American Journal of Psychology,* vol. 68, no. 4, Dec 1955. |
| [6] | I. Hsu, "LED brightness adjustment: high-frequency PWM dimming," Texas Instruments, Aug 2016. [Online]. Available: https://e2e.ti.com/blogs\_/b/powerhouse/archive/2016/08/26/led-brightness-adjustment-high-frequency-pwm-dimming. [Accessed Aug 2018]. |
| [7] | Y. H. &. H. L. J. Davis, "Humans perceive flicker artifacts at 500 Hz," *Scientific Reports,* vol. 5, Feb 2015. |

# APPENDIX