Lab 01: Blinky, Blinky, Blinky, and Clyde

ECE 484: Embedded Systems Design

Grade	Score	Signature
Attendance: James Nidda		
Attendance: Gabe Littecken		
Attendance:		
Demonstration	10	

Overview

In this lab, you will configure your USB AVR Serial Programmer (ASP) and install Microchip Studio to be able to flash programs to your ATmega328P microcontroller. After installing the necessary software, you will flash an example program to your microcontroller to interface with an LED and push button.

Deliverables

1. System Demonstration (10 points)
2. Completed Worksheet submitted to Canvas (5 points)
3. Commented Source Code submitted to Canvas (5 points)

Instructions

To begin the lab, navigate to the Canvas page and locate

ECE_484_USB_ASP_Config_Instructions.pdf

under the Lab 01 folder. This document will walk you through the necessary steps to configure Microchip Studio and your USB ASP programmer.

After successfully installing the software, create a new project in Microchip Studio for this lab. In the main.cpp file of your created project, copy and paste the example code found in the Lab 01 folder on Canvas. The example code is a simple program that constantly blinks an LED. The blink rate of the LED is dependent on whether a push button is pressed. If the push button is not pressed, the LED will blink at a slow rate. Conversely, when the push button is pressed and held down, the LED will blink at a faster rate. The example program constantly monitors the state of the button every instance through the while loop found in the main.cpp file. This method is referred to as polling and we will discuss the pitfalls of this method later in the class.

After you have copied the example program into your project, you will need to construct interfacing circuitry for the LED and the push button:

1. For the LED, you will need to include a current limiting resistor so that you do not exceed the 20mA current rating of the LED. To be safe, you will choose a resistance value such that the current through the LED is $\leq$ 10mA.

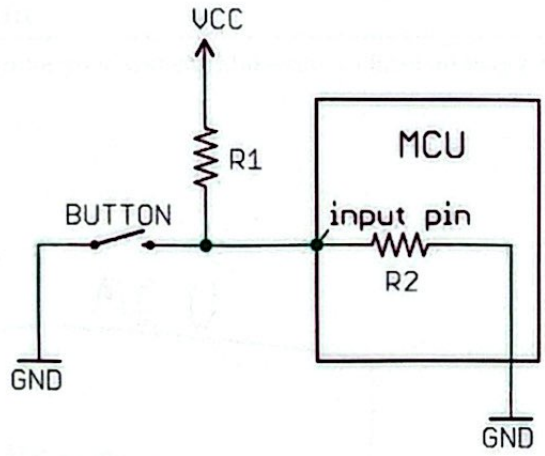


Figure 1: Pull-Up Resistor Circuit

2. For the push button, you will need to construct a pull-up circuit to hold the input pin high when the button is not pressed. The construction of a pull-up circuit will allow the input pin to see a reliable +5V when the button is not pressed and +0V when the button is pressed. As a general rule of thumb, a 10kΩ is a suitable pull-up resistance value.

Calculate the necessary resistance value for your LED current-limiting resistor and construct your circuits on your breadboard.

You are now ready to connect your circuit to your microcontroller. The pinout for the ATmega328P is available on the Canvas page. For this lab, the anode of the LED (and current limiting resistor) are connected to PIND7 while the middle node of the pull-up push button circuit is connected to PIND0. You will also need to ground your circuit using the GND pins and will source +5V using th 5V pin.

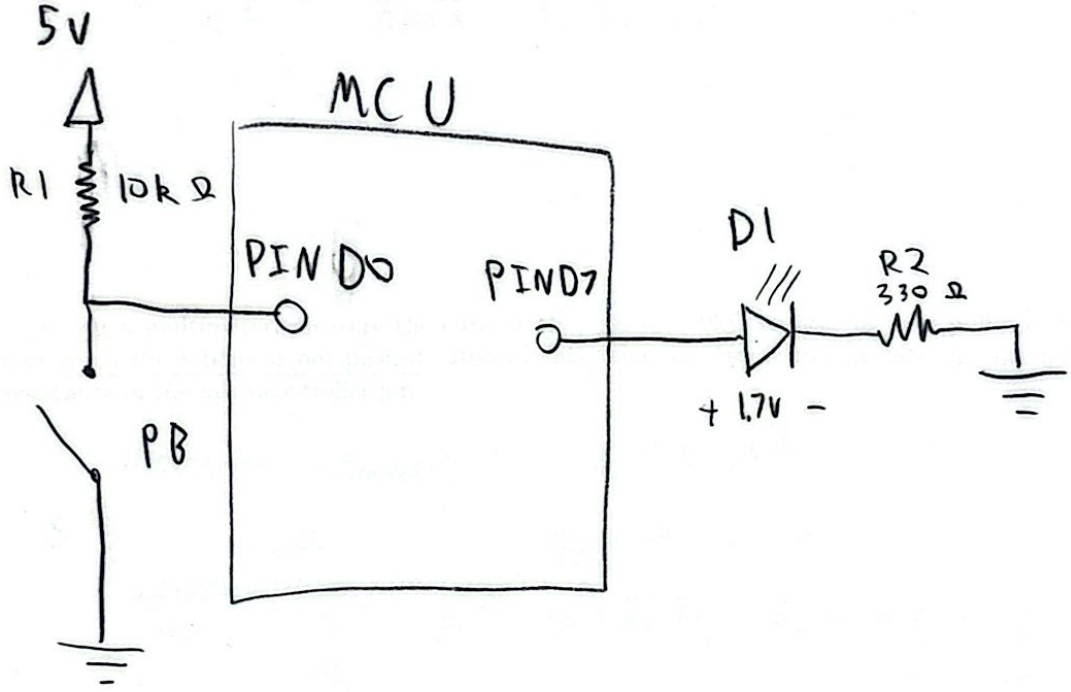
Demonstration Rubric

Please demonstrate your system to Dr. Swanson or the TA and they will record your demonstration score at the top of your lab manual. Checkpoint scores are provided in the following rubric. Scores between checkpoints may be awarded if a student has exceeded the previous checkpoint but has not reached the next checkpoint(s).

Score	Reasoning
10.0	The LED blinks slowly when the button is not pressed and faster when the button is pressed.
8.0	Student has partially constructed the interfacing circuitry but the LED does not blink or the button has no effect on the blink rate.
6.0	Student can program their microcontroller but has not constructed the interfacing circuitry.
4.0	Student has properly installed Microchip Studio but cannot program their microcontroller.
2.0	Student has ordered and received their lab kit, but has not properly installed Microchip Studio.
0.0	Student has not installed Microchip Studio and has not ordered/received their lab kit.

Circuit Diagram

Draw a circuit diagram for your system. Make sure to label all major modules, resistance values, and signal lines.



Design Questions

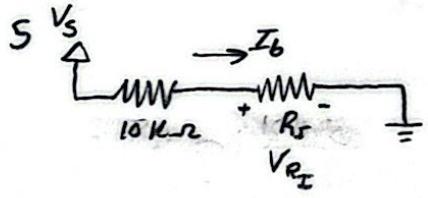
1. Show the calculations you used to select a current limiting resistance value.

arduino outputs 5 V...

$R = \frac{V}{I} = \frac{5V - 1.7V}{0.01\,A} = 330\,\Omega$

2. Using a multimeter, measure the current through the 10kΩ resistor in your pull-up circuit when the button is not pushed. Record this value and use it to calculate the internal resistance of the microcontroller pin.

Recorded current: 0.48 μA



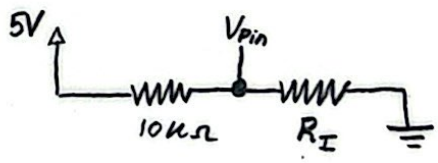
$R_{eq} = 10k\Omega + R_I$

$V_S = I_b R_{eq} = I_b(10k\Omega + R_I)$

$\rightarrow \frac{V_S}{I_b} - 10k\Omega = R_I$

$R_I = 10.41\,M\Omega$

3. Using the internal resistance value of the microcontroller pin calculated in question 2, calculate the voltage seen on the input pin (i.e. across the internal resistor) when the button is not pushed. What logic level does the calculated voltage value correspond to?



$V_{pin} = 5\frac{R_I}{10k\Omega + R_I} = 5\frac{10.4M\Omega}{10k\Omega + 10.4M\Omega}$

$= 4.995\ V$

4. When you press or release the button, there are instances where there is a perceivable lag in the changing blink rate. Why does this lag occur?

Idea 1: buttons are not debounced so additional code is needed to be run in order to detected the blink rate change.

Idea 2: In order for the status of pin 0 to be checked, the previous action needs to complete. When the button is pressed, the microcontroller will enter into a for loop for a certain amount of time. Same is true when the button is not pressed. It is necessary for the microcontroller to exit the for loop to read the state of pin 0 again.