Lab 02: What's All the Buzz About?

ECE 484: Embedded Systems Design

Grade	Score	Signature
Attendance: James Nippa		Ja
Attendance: Gabe Litican		
Attendance:		
Demonstration	10	

Overview

In this lab, you will design an embedded system that will count the number of button presses and display the count on a single seven-segment display. The designed system will also incorporate an active buzzer that will serve as a 4-bit overflfow warning.

Deliverables

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

Instructions

Your system will consist of a push-button, active buzzer, and a seven-segment display. Each segment of the seven-segment display contains an LED that will need a current limiting resistor to interface with the microcontroller. A 330Ω resistor is a suitable resistor value. You will need to construct seven LED circuits that connect to the seven appropriate pins on the seven-segment display, as shown in the Figure 1. You will connect the LED circuits sequentially to PIND0-PIND6 with seven-segment pin A being connected to PIND6 and pin G being connected to PIND0.

After connecting your seven-segment display, you will need to choose two other GPIO pins for your push button and active buzzer. The push button requires a pull-up circuit (a 10kΩ resistor will suffice) and you will connect the middle node of the pull-up circuit to a GPIO input. For the active buzzer, you will connect a GPIO output to the anode (long arm) of the buzzer and connect the cathode (short arm) to GND.

For the software, you will need to write a program that first polls the push-button input and detects a **falling edge**. Remember, the input signal is held at a logic high when the button is not pressed and a logic low when the button is pressed. Once a falling edge is detected, a count variable should be incremented You will then check if there is a 4-bit overflow and reset the count as necessary or sound the buzzer if the 4-bit count is about to overflow. After these checks, you will call the `setSevenSegments` function to output the count variable on the seven-segment display. Starter code containing the `setSevenSegments` function is posted on Canvas.

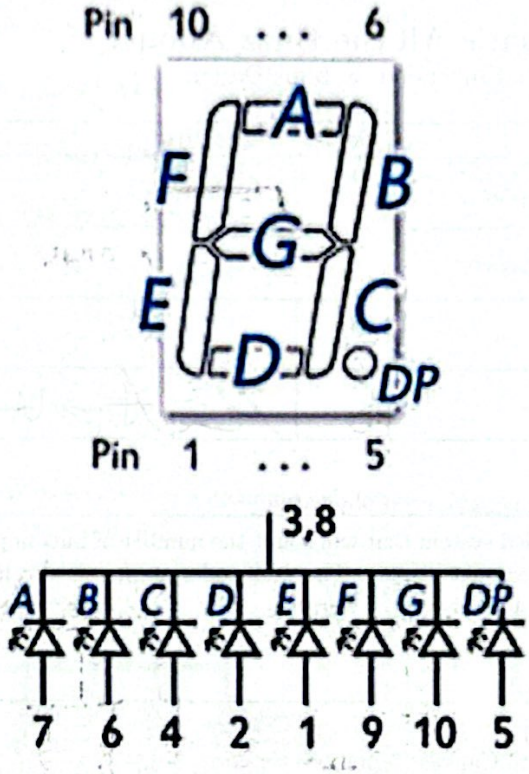


Figure 1: Common Cathode Seven Segment Pinout

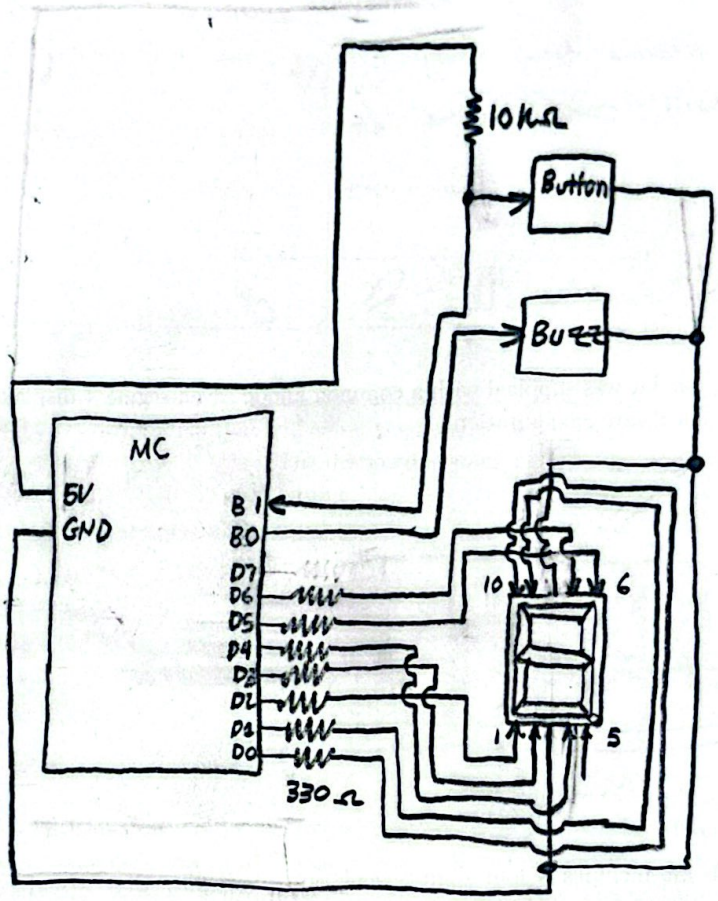
Demonstration Rubric

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 student properly increments the count and displays it on the seven-segment display. The count overflows properly and the buzzer activates when the count is about to overflow.
8.0	Student can properly display the current count on the seven-segment display, but the count increments incorrectly when the button is held or released. The count also does not overflow properly and the buzzer does not activate when the count is about to overflow.
6.0	Student can change the seven-segment display by changing the GPIO signal, but cannot properly display the current count
4.0	Student can light up the seven-segment display with +5V and GND but cannot interface with GPIO.
2.0	Student can program a simple example program to their microcontroller.
0.0	Student cannot program their microcontroller and has not installed microchip studio.

Circuit Diagram

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

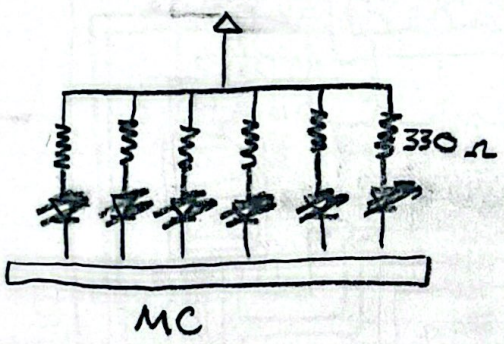


Design Questions

1. When the button is pressed, there are instances where the count increments by more than 1. Why do you think this happens?

The button that we are using is not debounced. This means that when it is released from being pressed, the button has the potential to bounce. This will complete the circuit which is why we see this behavior.

2. Suppose your kit was supplied with a common anode seven-segment display. How would this change your hardware configuration?



Pins 3 & 8 would be connected to VCC. The resisters would remain in the same place.

3. Your lab kit includes a four digit seven-segment display that would require 28 GPIO pins to control if you repeated the hardware configuration in this lab for each of the displays. 28 GPIO pins greatly exceeds the number of GPIO pins available on your microcontroller. How could you control the four digit seven-segment display using your current microcontroller?

Because the clock of the board is 16 MHz, We can simply cycle through each display and display the digits separately. for this to work 3 or 4 pins will be needed to switch btw the displays. the remaining pins will control what is displayed.