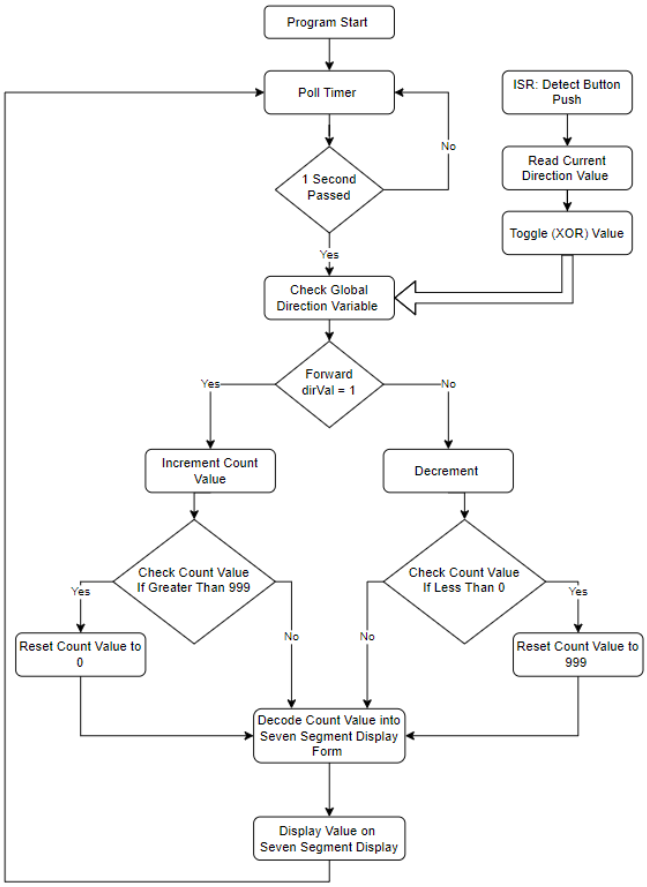
ECE 9047: Lab 2

Submitted By: Joshua Bainbridge 250869629

Part 1: Problem statement

The goal of this lab is to implement the following system: Create a three-digit decimal counter that accurately counts 1 s intervals. The count should be displayed as a decimal number on the 7-segment display. An interrupt-enabled push button should be used to reverse the direction of the counter.

Part 2: Outline of Solution



The flow chart above give a complete overview of the solution used to solve the problem statement given in Part 1. Firstly the ISR responds when a button is pressed on the microcontroller. The button implies a change in direction of the counter. A global variable will be used so both the ISR and a main program can have access to the information. The timer will be polled to determine when a second has elapsed. Once a second has passed the global direction variable is read to determine whether to increment or decrement the count value. After the count value is incremented or decremented the program tests to see if the new value is bout of bounds i.e. a vale of greater than 999 or less than 0. If this occurs the program resets the value of the counter to create a loop. Finally the value is sent to the

decoder coding used in Lab 1 and then displayed on the seven-segment display. The program will return to polling the timer until another second has passed.

Part 3: Approach to Solving Problem

Before starting to solve the problem statement, the given codes were first tested to gain a thorough understanding of how both the timer and interrupt programs work. The interrupt program was tested first. After looking at the code, it was clear the majority of the program can be used in the final code. The interrupts and buttons are responsible for altering the direction the counter counts. This was implemented by using a simple bit flip, stored in a global variable. This was implemented into the interrupt program. The bit was flipped using the bitwise XOR function (EOR). The function of the program was tested by running the program and watching the memory location of the global variable.

The next step was to repeat the first step with the timer. The code used in the lessons was copied to act as a starting point. It was noticed the code in the lab loaded the wrong values into the timer. Once this was corrected the program worked as described. It was clear this program could also easily be altered to solve the problem statement. The one second interval and five second repeat timer signal were kept how ever the LED program was replaced with the counting code. Register 5 was assigned to be the counting variable. It was selected because the timer program does not require the register. The counter was constructed in steps. The max value was set to 15 and the increase counter was made, followed by the decreasing counter, and finally the wrap around condition was implemented.

The next step was to combine the timer code and the decoding/ seven segment display program used in the full solution of Lab 1. Due to the fact the code used in lab one was very inefficient, so memory management was required. When the display and seven segment display function is called the system pushes registers 0 – 4 to the stack. This frees up enough space for to run the code. Before returning to the timer polling code the stack is popped back.

Finally the two separate programs were combined and tested. To ensure the program was properly working the count, the global variable and the LEDs were monitored. Once the program the count and buttons were confirmed to work the LEDs and other overhead and testing codes were removes to produce the final program.

Part 4: Cost Benefit Analysis

The cost of designing the solution and implementing the program using the method described in section 3 was pretty low. Testing the timer program before attempting to solve the problem statement saved a lot off the overall design. Additionally, implementing three separates prewritten programs also save a lot of time. This it because it was known the code works and simply had to be modified to solve the problem statement.

Overall the full process to solve the problem statement was roughly one day's work, including designing and debugging the given timer code. Additional time was required to fix memory issues and cobbled registers in the fully combined iterations. This compared to what I imagine would take many day to understand and write the code for interrupt and timer.