**EGRE 364 – Microcomputer System**

**Laboratory Number 4**

**Up-Down Counter in Assembly**

**Lab Section: Tuesday 1PM**

***Lab conducted on: September 24, 2019***

***Report Submitted on: October 11, 2018***

**Papa Beye & Omar Amr**

**Major: CPE & CPE**

**Introduction**

The purpose of this lab was to build on the concepts of the use of multiple GPIO input and output configurations and perform digital I/O input and output programming using Assembly in the Keil µVision software development environment. These configurations include registers input/output, push pull, open drain, pull up/down, ODR, IDR. The objective of this lab was using the new 7 segment display to create an up-down counter that interfaced with microcontroller to be controlled as a counter with the joystick on the board and completing this objective by using Assembly programming. We used branches and mnemonics such as STR or BIC. We were able to initialize each GPIO port and manipulate the appropriate ODR and IDR values to light up the LED based on the status of the counter. We were given the truth table of the 7-segment display for numbers 0-9 and A-E. We were able to light up the specific pin for each segment to display each number by using PE10 – PE15 and PH0 with a common cathode connection. We used the PA3 and PA5 input pins to input the up and down to increase or decrease the displayed value. We initialized the PA3 and PA5 pins as pull down and we had to debounce the button input with software. We needed to debounce the mechanical input so that we make sure that the number displayed only changes only one per button press. We used software debouncing in our assembly code by waiting until there was constant input for a given amount of time and set the input to that value, thus allowing a time delay until the input is constant and debouncing the input. Our Assembly program checks for button press and button release to allow the mechanical switch to be debounced. In this lab, we used the practice and methods that we learned in class and overall processes from lab 3.

**Functionality and Correctness**

The STM32L4 discovery board has a joystick on board with 5 input buttons. In this lab, we used the up and down push buttons, PA3 and PA5. Since both inputs’ buttons don’t have an internal pull-down resistor, we must initialize the pins as pull down. Since the joystick is a mechanical system, we must use software debouncing to stop the mechanical bouncing when the button is pressed. We used a simple routine delayed for certain time while the input is constant and afterwards sets the value of the input. The assembly code below shows how we were able to wait until the input stayed constant to increment or decrement the value of the LED. The seven segment display is made up of ten pins and we need to use pins PE10-PE15 and PH0 on the board to activate a certain segment by setting the respective bit in GPIO port E and port H; Using the given truth table of the 7-segment display, we were able to create a routine that would set the appropriate ODR bit(s) so that the display showed the requested number. In order to “roll over” when incrementing past 9 or decrementing past 0, we had to use condition branch that made sure that if the incrementing “count” variable was greater than 9, we reset it to zero. Likewise, if the “count” variable was less than 0, we reset it to nine.

**Assembly code showing implementation of debouncing**

    LDR r0, =GPIOA\_BASE

    LDR r1, [r0, #GPIO\_IDR]

    LDR r2, =0x28

    AND r1, r1, r2

    CMP r1,#0x8

    LDR r0,[sp,#0x08]

    BEQ A3bd

    BNE ct5

ct5 LDR r0, =GPIOA\_BASE

    LDR r1, [r0, #GPIO\_IDR]

    LDR r2, =0x28

    AND r1, r1, r2

    CMP r1,#0x20

    LDR r0,[sp,#0x08]

    BEQ A5bd

    BNE st

A5bd  CMP r0, #0x0

      BEQ A5check

      BNE A5dbc

A5dbc SUB r0, r0, #0x1

      B A5bd

A5check LDR r0, =GPIOA\_BASE

        LDR r1, [r0, #GPIO\_IDR]

        LDR r2, =0x28

        AND r1, r1, r2

        CMP r1,#0x20

        BEQ A5check

        BNE dec

A3bd  CMP r0, #0x0

      BEQ A3check

      BNE A3dbc

A3dbc SUB r0, r0, #0x1

      B A3bd

A3check LDR r0, =GPIOA\_BASE

        LDR r1, [r0, #GPIO\_IDR]

        LDR r2, =0x28

        AND r1, r1, r2

        CMP r1,#0x8

        BEQ A3check

        BNE inc

inc LDR r0,[sp,#0x04]

    ADD r0, r0 ,#0x1

    STR r0, [sp,#0x04]

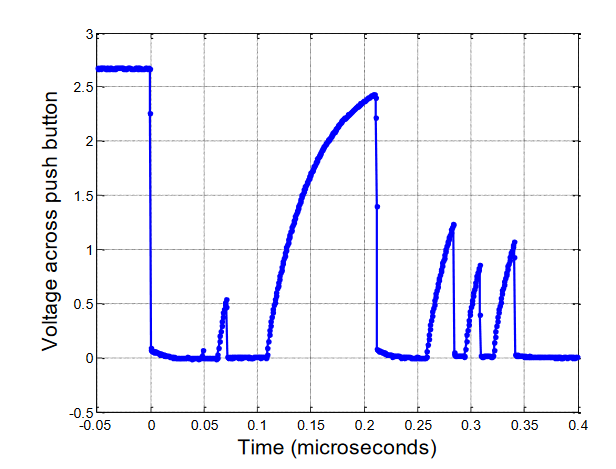
    B disp

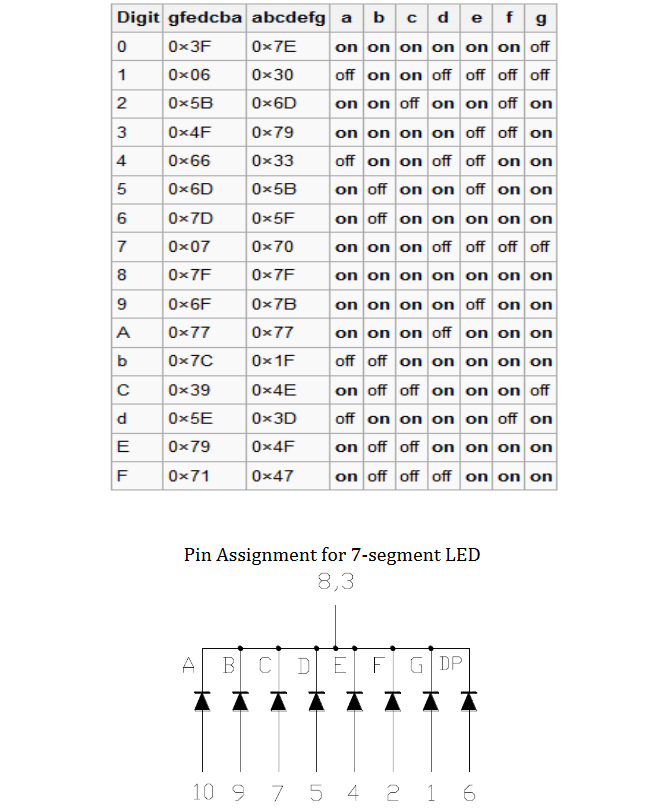
dec LDR r0,[sp,#0x04]

    SUB r0, r0 ,#0x1

    STR r0, [sp,#0x04]

    B disp





**Pre-Lab & Post Lab included below**

**Conclusion**

In this lab, we became more comfortable with interfacing with a seven segment LED, implementing inputs, and correctly debouncing inputs with software using Assembly programming. We learned about checking the IDR value of a pin and debouncing the input using branches. We were able to achieve the objective of the up-down counter using the joystick and 7-segment display with proper software debouncing. We used branches and CMP statements to be able to test and run routines to receive inputs from the up and down buttons and alter the LED value. We learned a lot about how to use branches and Assembly mnemonics and labels to initialize ports, access registers and reading inputs with debouncing. We didn’t face any issues but there was a learning curve that allowed us to learn how to create and use Assembly branches to properly access and manipulate bit values of interest without altering other bits and reading inputs through the IDR.