**ESET 269**
**Embedded System Development in C**

Spring 2023
Final Laboratory Report

# LED Passcode Indicator Using The MSP430 Microcontroller

Author's Name: Mark Perez

Lab Due Date: May 4, 2023

Lab Submitted Date: May 1, 2023

**IS THIS LAB LATE?** No

All of the information contained in this report is my own work that I completed as part of this lab assignment. I have not used results or content from any other sources or students.

Mark Perez
Printed Name

Mark Perez
Electronic Signature

## Motivation:

The motivation by which this project was created was to gain experience in using microcontrollers embedded with C programming skills while integrating these concepts using the application of a real-world scenario. In this project, we incorporated our programming skills in order to develop a mock passcode entry system using a set of buttons and LEDs on the MSP430 microcontroller. In doing so, we will be able to fully assess the concepts we have been taught in relation to both our programming skills and understanding of the microcontroller, which are both skills necessary for our future courses and careers.

## Methods:

In the program for this passcode entry system, we used various methods both involving the basic programming techniques in regards to programming in C language as well as the programming methods involved in the function of the MSP430 microcontroller. Before determining the methods we will use in order to create the program for this project, we first had to create a structure to base our code on, which we used the design of a finite state machine to do. In doing this, we were able to map out each of the states that the program would encounter from the start of the program to the end, which was the final state of the correct combination. In regards to the methods involving the basic programming in C language we used in this final project, we used a combination of while loops, if-else statements, and switch-break case statements in order to create the functional code for this mock passcode system program. The while loop was used in order to create an infinite loop that would allow for the program that was created in order to implement the passcode system to run infinitely. The if-else statements were used in order to read the inputs from the buttons and issue the numbers to each of the switches. Lastly, the switch-break case statements were used in order to implement the state machine that was designed, in which the state machine diagram can be found below. This switch-break case is the foundation of the program we created since it ensures that the system functions correctly and only opens with the right combination of buttons pressed. In regards to the programming techniques that were used in order to create the functional passcode system code, we used the directory (PxDIR) and output (PxOUT) statements in order to declare the switches and LEDs we were using on the board of the microcontroller as well as the resistor enabling (PxREN) function, which will allow for us to enable the resistor as being pull-up or pull-down. These three statements were then paired with the use of the logical operators of the AND (&), OR (|), and NOT (~) gates, which allowed us to control several aspects of the microcontroller. The combination of these various methods allowed for us to create the working program that would act as a basic passcode entry system.

## Program:

In order to fully understand the coded program for the password entry system that was created, we must first discuss the design for the finite state machine, in which this setup is the

base on which our code will be constructed from. Below you will find the constructed state machine diagram, which shows the various paths that the user may face when using the final product. As you can see from the diagram shown below, the correct combination of inputs from the button should be 2-1-1-2. The state machine diagram not only shows the correct input combination in order to "open" the lock, but it also shows if an incorrect input is made at any point in the sequence, the sequence is always reset back to its original state. As a quick side-note, this combination was chosen based on an album by the rock band Rush, which happens to be my father's favorite band and a known password in my family as I have been growing up.

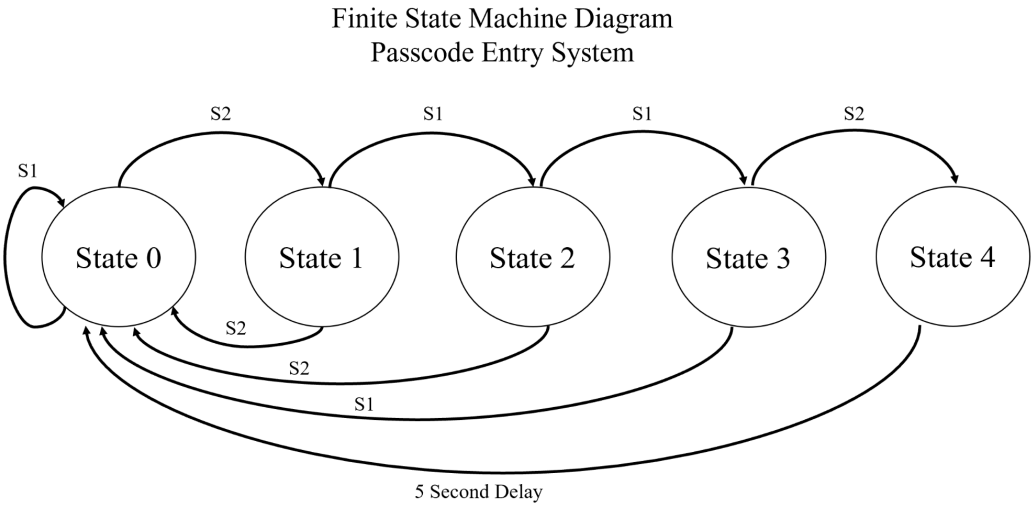


***Figure 1: Finite State Machine for Passcode Entry System***

Next, we will be discussing the code that was developed in order to allow for the microcontroller to function as the "password entry system" we intend to develop. In the first block of code shown below, we can see the use of the directory, resistor enabling, and output statements in order to distinguish the elements we are going to be using from the microcontroller board. It is important to add that for this circuit, we will be using pull-up resistors which produce a high value when the button is not pressed and a low value when the button is pressed. As shown on the board and in the code, we used the 1.1 and 2.1 switches as the inputs for our design as well as the 1.1 and 4.7 LEDs on the microcontroller, which act as the output showing whether or not the correct input combination has been entered through the flashing of either a red or green LED. Along with the declaration of the switches and LEDs, we also see the implementation of the while loop, which will allow this password system to run indefinitely. The code shown below also includes the if-else statements that would allow for us to read in the inputs from the switches and relay that information to the code that would determine whether or not the correct combination has been entered. The first if statement indicated that if both PIN 1 and BIT 1 produce a high value and PIN 2 and BIT1 are equal to zero, then button 2.1 is active and is assigned as Switch 1. The second if statement indicates the vice versa in order to assign

button 1.1 from the microcontroller as Switch 2, which would allow us to implement the state machine.

```
#include <msp430.h>
int state = 0; // Global Variable //

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;   // stop watchdog timer

    // Switch 1 //
    P2DIR &= ~BIT1; // Push Button 2.1 As Input 1 //
    P2REN |= BIT1; // Enabled Resistor //
    P2OUT |= BIT1; // Pull-Up Resistor //

    // Switch 2 //
    P1DIR &= ~BIT1; // Push Button 1.1 As Input 2 //
    P1REN |= BIT1; // Enabled Resistor //
    P1OUT |= BIT1; // Pull-Up Resistor //

    // LED Declaration //
    P1DIR |= BIT0;
    P4DIR |= BIT7;

    // Initial States //
    P1OUT |= BIT0;
    P4OUT &= ~BIT7;

    int Switch = 0;

    while(1)
    {
        // Read Inputs From Buttons //
        if(((P1IN & BIT1) != 0) && ((P2IN & BIT1) == 0))
        {
            P1OUT &= ~BIT0;
            __delay_cycles(300000);
            Switch = 1;
        }
        else if(((P1IN & BIT1) == 0) && ((P2IN & BIT1) != 0))
        {
            P1OUT &= ~BIT0;
            __delay_cycles(300000);
            Switch = 2;
        }
        else
            Switch = 0;
```

***Figure 2: Password Entry System Program Part 1***

In the second portion of code for this program shown below, we will find the switch-break statement portion of this code, which is within the while loop and will run indefinitely just as the previous if-else statement. In this switch-break statement, we have five separate cases, which looking back to the finite state machine diagram represent the five separate states that we must integrate in order to correctly design the program. For the first four cases, which are represented as states 0 through 3, the result from each of these states does not result in the complete correct combination, so the red LED, or LED 1.0, will be the LED that is ON in these states. In each of these cases, we use if-else statements, which act as the arrows in our state machine diagram, that allow for the flow between each case. If the correct switch is activated, then the counter for the states is incremented by 1, and if the incorrect switch is activated, the program returns to the original state, which is State 0 or Case 0. Once the program reaches case 4, the correct combination has been entered and LED 1.1, which is the green LED, is now activated while LED 1.0 is deactivated. We also use the delay cycles function in order to allow for the LED to turn on for 5 seconds once the correct combination has been entered, and then the program returns back to its original state of state zero in order to have the user have multiple attempts at the program while it is running.

```
        // Finite State Machine //
        switch(state){
            case 0:
                P1OUT |= BIT0;
                P4OUT &= ~BIT7;
                if (Switch == 1)
                    state = 0;
                else if (Switch == 2)
                    state++;
                break;
            case 1:
                P1OUT |= BIT0;
                P4OUT &= ~BIT7;
                if (Switch == 1)
                    state++;
                else if (Switch == 2)
                    state = 0;
                break;
            case 2:
                P1OUT |= BIT0;
                P4OUT &= ~BIT7;
                if (Switch == 1)
                    state++;
                else if (Switch == 2)
                    state = 0;
                break;
            case 3:
                P1OUT |= BIT0;
                P4OUT &= ~BIT7;
                if (Switch == 1)
                    state = 0;
                else if (Switch == 2)
                    state++;
                break;
            case 4:
                P1OUT &= ~BIT0;
                P4OUT |= BIT7;
                __delay_cycles(5000000);
                state = 0;
                break;
        }
    }
    return 0;
}
```

***Figure 3: Password Entry System Program Part 2***

## Results:

In order to view the results of the completed password entry system, I have provided a link below, which will allow for you to view the functionality of the program that was created for this final project. As you can see from the video provided, each wrong combination produces a flash of the red LED until the correct combination has been entered. Once the correct combination has been entered, the green LED is activated for five seconds before returning to the original state to allow for the process to continue. The link to the video showing the result of this program that was created is as follows: https://www.youtube.com/watch?v=dr9_YdFSANw . As shown in the video, the first two combinations did not result in the correct combination, and the red light was the only output shown until the correct combination was entered which occurred as the third trial. The results from this video show how the program we have designed is successful in its functionality, and we can verify this by testing the program when run on the microcontroller based on the state machine diagram we have created.

## Conclusion:

All in all, the completion of this final project has allowed for us as students to be able to solidify our knowledge on the topics we were taught in this course. In this particular project, we were able to tie in multiple topics learned in the programming language of C with our newfound knowledge of the programming style of the MSP430 microcontroller. In merging these two

important concepts together, we not only were able to test our knowledge on the concepts we had been learning, but we did so by integrating those concepts using a tool that is commonly used in jobs we may obtain in the future. Knowing this, we can see the impact this final project has on students since it allows for us to obtain practical experience using the knowledge gained in this course with components we are bound to see when entering the job force.

Along with the positive impacts this project has had, I feel that this project can be expanded in order to further the depth of knowledge in these concepts. I feel that if there was more time for this project, I would have added more features into this program such as the option for a user to input a combination and have other users guess the combination. I would also want to add the possibility for the user to have limited time before "locking" them out for varying amounts of time depending on the amount of tries the user has used. This would be beneficial since it would relate to options found in the real-world such as the password functions on a cellphone.

As mentioned before, this project definitely has an impact on my future, in regards to both future courses and career-wise, since it allowed not only for the gain of knowledge in programming in yet another programming language, but it also integrated the programming with both real-world scenarios, applications, and components. The sheer fact of having a great foundation and understanding of programming in multiple languages, especially C since it is more widely used in common applications, shows how gaining the knowledge from this final project has an immense impact on our future. In having a successful, completed project, we are able to take forth with us the knowledge gained in this course into the workforce when we graduate from this institution.

## Statements of Encouragement:

As a statement of encouragement to future students of the course, I would like to say that the best way to ensure your success on this project as well as in the course itself is to take the homework assignments seriously and practice coding as much as you can. In a course like this, the saying "practice makes perfect" holds true in this course, and in gaining a strong hold on these concepts allows for an easier transition into both programming the microcontroller as well as the completion of the final project. Lastly, I would like to say that although many concepts taught in this course may seem very tedious or difficult, there is always a solution to any problem that you are faced with, and the idea of "never giving up" also holds true for this course.