16/20

Project 1 - Hello World Electronic Lock

Students: Zach Bunce & Garrett Maxon
Class: CPE 329-03
Quarter: Spring 2018
Date Submitted: 4/20/18
Professor: Gerfen, Jeffrey

Secret Sauce

At any point, the user can enter " #*" to enter a privacy mode which does not display the entered combo onto the screen while still maintaining all locking, unlocking and clearing functionality. The user can also exit the privacy mode by entering "#*" again. In addition to the privacy mode, the unlock screen was given an animation update that makes the unlock text "HELLO WORLD" scroll across the screen and then to the next row when the edge is hit.

Purpose

The purpose of the project is to design and code a working electronic lock that integrates a LCD module and a 12-key keypad. When the correct combination is entered, the LCD will display "Hello World" and when the incorrect combination is entered it will display "Incorrect Code". In the process of doing this students learned how to integrate peripherals with the MSP432.

System Requirements

- At power on, the LCD will display the lock screen and ask the user for a passcode.
- Each key press in an attempt will be displayed on the LCD.
- The system will accept a single four digit passcode before resetting.
- When a combination is entered, the LCD must display the corresponding message.
- When the asterisk key (*) is pressed, the previously entered keys shall be cleared.

Youtube Video Demonstration

https://youtu.be/JRE7xoLy8Jw

Passcode: 9137

-1
Organize the system requirements by listing the main points and then sub points below. e.g.
- The system shall have an LCD screen
    - the LCD shall have a screen of 16x2 characters
-etc…

System Specifications

Table 1: System Specifications

Hardware	LCD NHD-0216HZ-FSW-FBW-33V3C
Dimensions	14mm/36.7mm/65.5mm H/W/L
Character Resolution	2x16 R/C
Operating Voltage	3.3V
Startup Time	45.6316 ms
Response Time	474 µs
Hardware	Texas Instruments MSP432 Microprocessor
I/O Pins Used	14
Operating Voltage	3.3V
Clock Frequency	48MHz
Hardware	Keypad
Key Count	12
Dimensions	12mm/51mm/70mm H/W/L
Hardware	Overall System
Dimensions	4.5”/6.5”/6” H/W/L

OK

System Architecture

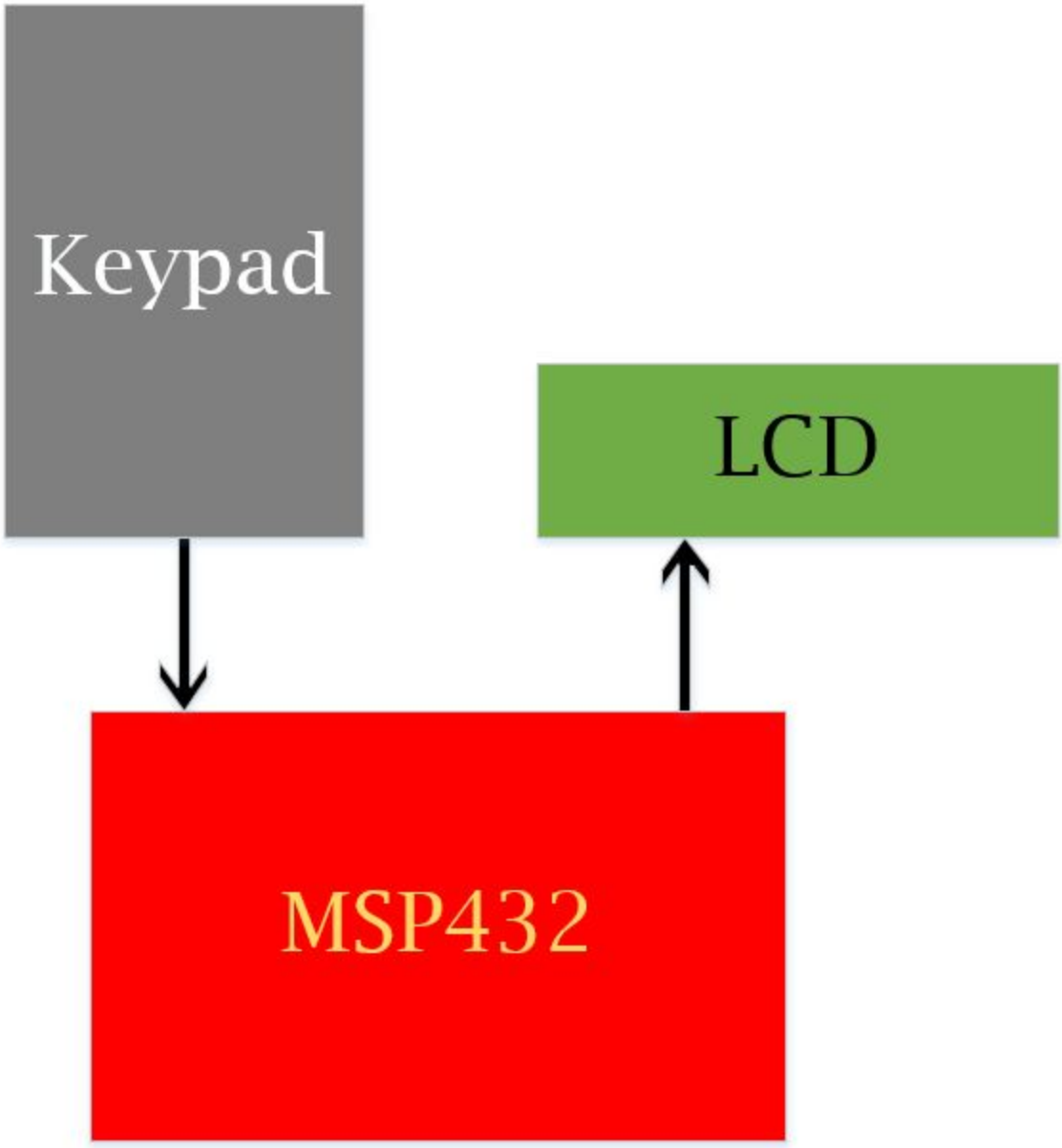

Figure 1: High Level System Block Diagram

Component Design

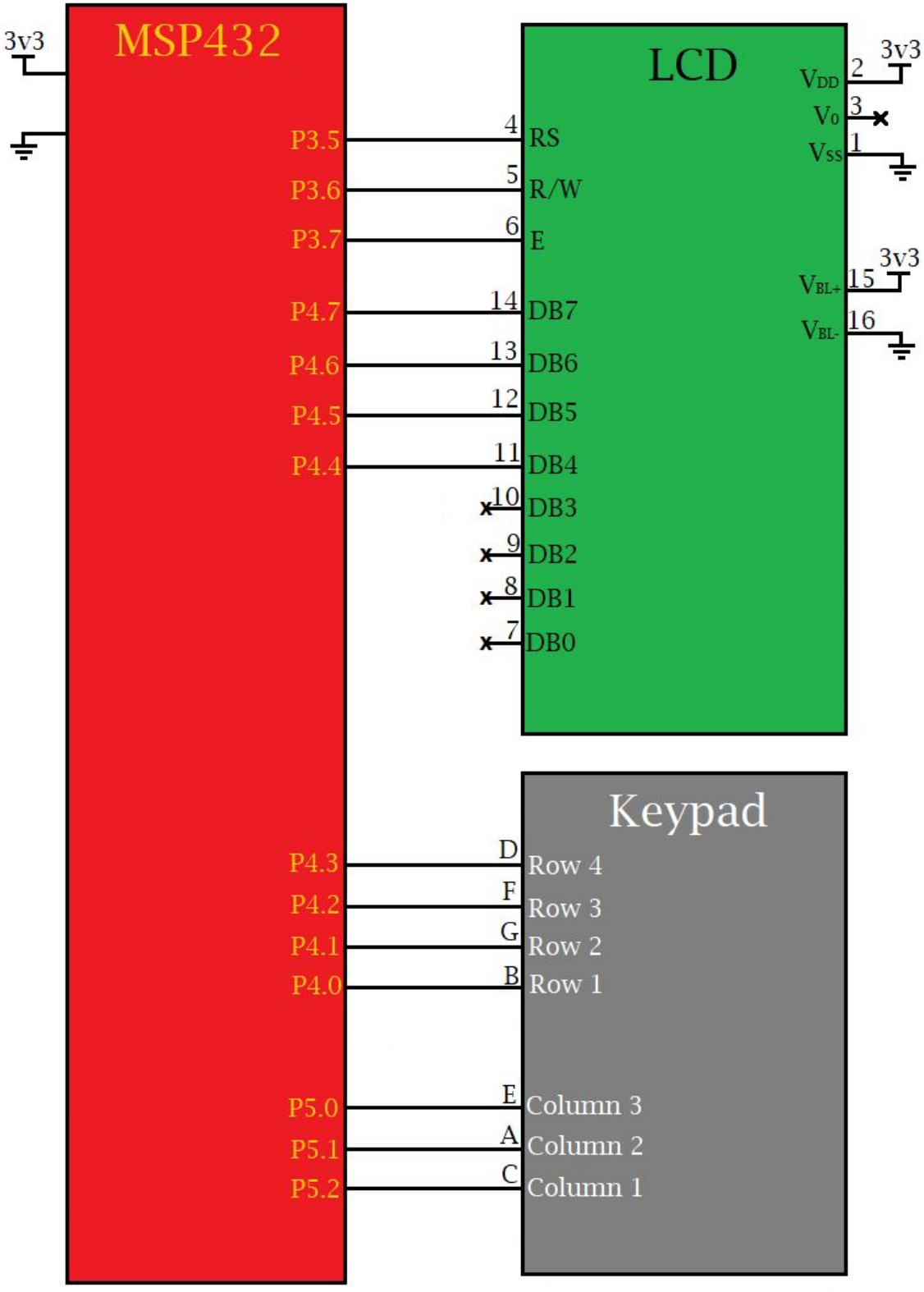

Figure 2: Schematic diagram of complete system including MSP432, LCD, and keypad

-.5
Include pin numbers on the MSP432

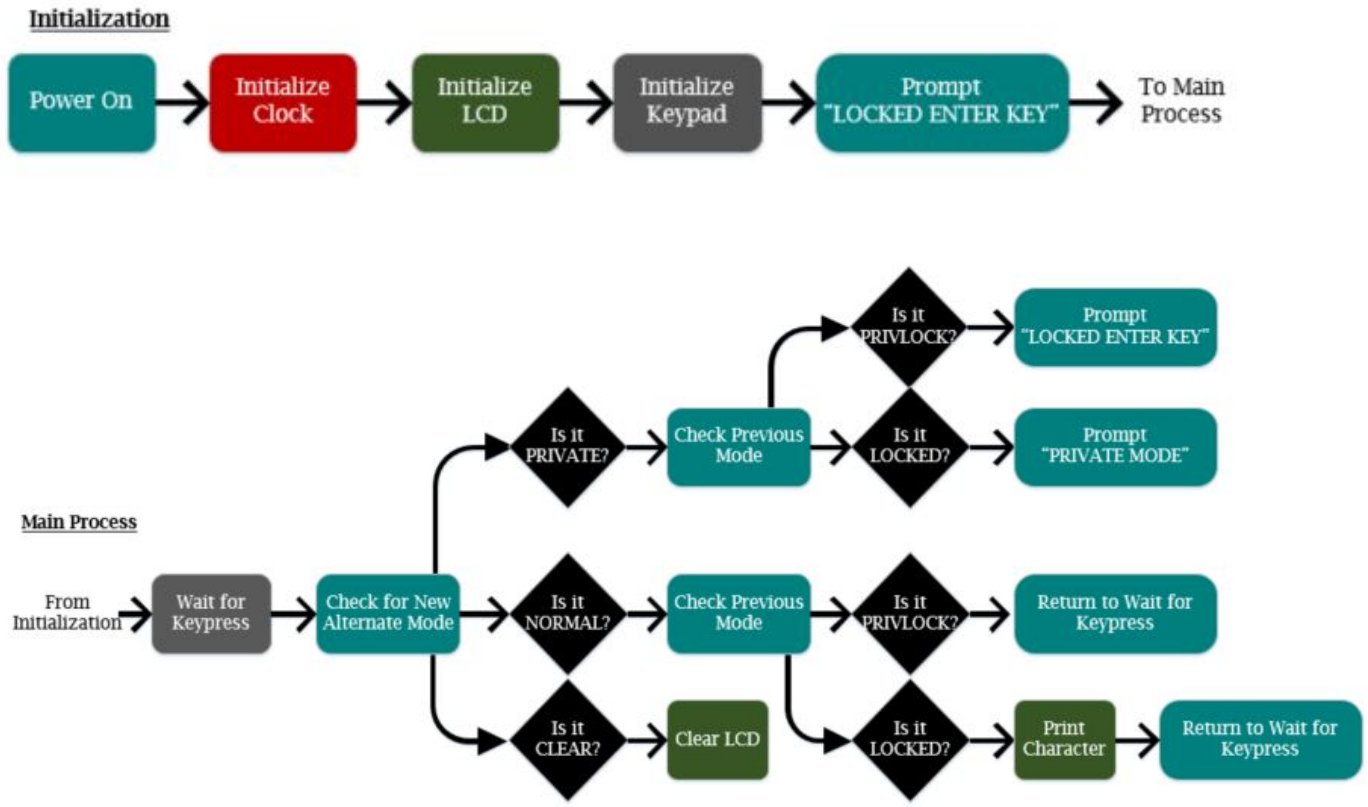

Figure 3: Process Flow Diagram

Bill Of Materials

Table 2: Bill of Materials

#	Part Description	Part Number	Distributor	Quantity	Price
1	MSP432 Launchpad	MSP-EXP432P401R	Digikey	1	$13.49
2	LCD Module	NHD-0216HZ-FSW-FBW-33V3C	Digikey	1	$11.60
3	12 Key Switch Keypad	COM-08653-ND	Digi-Key	1	$3.95
4	6" M/M & M/F Jumpers (Strip of 10)	DZ-DBX-01	Amazon	1	$6.98
5	25 Header Pin Strip	78511-436HLF	Digi-Key	1	$1.38
6	Breadboard 830 Point (3 pieces)	EL-CP-003	Amazon	1	$9.99
				TOTAL	**$47.35**

System Integration

In order to construct the complete system, the individual peripheral libraries were first constructed and tested separately over the course of Assignments 1-4. As Assignment 4 required the integration of the keypad and LCD, both of which relied on earlier libraries such as the delays and clock setting, the behavior and interaction of all of the libraries used had already been verified before the design of this project began. Thus, the design process of the project began by interpreting each project specification into sequential processes and adding to these processes those required by the extra specification determined by the designers. From this analysis, it was determined that the system would need to first display the locked message onto the LCD after initialization, wait for four key presses, and then evaluate the system's lock state. It was also determined that the system must also check against a few predetermined input combinations while accepting key presses in order to know whether it needed to clear the digits entered, or enter/exit the designer-added specification of privacy mode. With these large processes now interpreted, each one was analyzed in order to locate similar sub-processes which could be generalized and combined together to increase code conciseness. Thus, at the system level, it was determined that five functions would be required: obtain a key press, check the mode, check the password, set up the lock screen, and set up the unlock screen.

As both the necessary system processes and sub-processes had been determined, development began on constructing the actual C functions to implement these processes. The complete code written for each of these functions, as well as the other libraries used, was recorded below in ***Appendix A***. During the implementation process, the original libraries constructed in previous assignments were found to be lacking, and as such additional functions were written and tested for them, such as *write_string_LCD* and *line_clear_LCD* for the LCD library, shown below in ***Figure 4*** and ***Figure 5***.

```
//Takes in DDRAM address pixel and ASCII string word
//Word wraps if line ends are reached
void write_string_LCD(char word[], uint8_t pixel, int CLK)
{
    uint8_t i;
    uint8_t location = pixel;
    uint8_t len = strlen(word);
    for(i = 0; i < len; i++)
    {
        if ((location > 0x0F) && (location < 0x40)) {
            location = 0x40;
        }
        else if (location > 0x4F) {
            location = 0x00;
        }
        write_char_LCD(word[i], location, CLK);
        location++;
    }
}
```

Figure 4: LCD.c function which writes a complete string to the LCD

```
//Clears the specified line of the LCD and puts the cursor on the next line
void line_clear_LCD(int line, int CLK)
{
    char blank[] = "                ";
    if (line == TOP) {
        write_string_LCD(blank, 0x00, CLK);
        home_LCD(CLK);
    }
    else if (line == BOTTOM) {
        write_string_LCD(blank, 0x40, CLK);
        write_char_LCD(0x10, 0x3F, CLK);
    }
}
```

Figure 5: LCD.c function which clears a single row on the LCD

After adding these functions to the LCD library and thoroughly testing them, the system implementation continued. Due to basic nature of the project, and the analysis performed before beginning the code implementation, no further major design changes were required, and the main portion of the project was completed with little issue.

However, there were a few major issues encountered after the basic version of the lock was completed. These problems were encountered during implementation of the extra mode management. These issues included not being able to clear the input line while checking for privacy mode, not reading the keypad input while in privacy mode, and problems with entering and leaving privacy mode. As most of these issues were related to the scope and data flow through sub-functions, since under certain conditions multiple modes could occur at once, some of the issues were fixed by simply modifying the functions involved in order to pass more information between them. Furthermore, mode memory was added so that the functions could track the current mode, not just the next mode, ensuring that the proper protocol would occur when attempting to enter or exit privacy mode.

Conclusion

This project was a good introduction to interfacing peripherals with the MSP432. During the overall process of interfacing the LCD and keypad throughout the various assignments, as well as constructing the other libraries used such as the delays, many important and not immediately clear details were found that were necessary to ensure proper integration of each peripheral. For example, operating the LCD in 4-bit mode required a fairly strict initialization process, as well as some basic serial data transmission. For instance, when initializing the LCD in 4-bit mode, three identical wake up commands must be sent to it with specific timings between them. These commands consisted of writing 0x30 to the instruction register, with 160µs between each command. This was not noted anywhere in the datasheet except the example code provided, causing some issues before this procedure was found. The 4-bit mode operation also caused some issues once keypad integration was attempted, as the partial register not used by the LCD was recycled for the keypad. However, due to a lack of bit masking in the LCD library, many errors were initially encountered. *This served as a useful reminder as to the reason for paying attention to proper masking procedures, even when writing code that would not require it on its own. One improvement that would be useful to implement is a debounce function when watching for keypad presses in order to avoid any issues, and to then use less arbitrary software delays which would smooth out operation. Another change that would improve the project's general usability is to have the possibility to change the passcode for future operation once the device is unlocked.*

-1
Needs 3 separate concluding statements in italics. These should be 3 important takeaways that will help you in future projects.

Appendix A: C Code

Main C Function

```
/**
 * main.c
 *
 * Locks the LCD until the correct sequence of keys is pressed
 * Once the correct sequence is pressed, displays "HELLO WORLD" unlock message
 * Allows for the previously entered keys to be cleared with "*"
 * Contains a privacy mode in which key presses are not printed
 * Privacy mode is both entered and exited with "#*"
 * Clearing is also enabled while in privacy mode
 *
 * Date: April 13 2018
 * Authors: Zach Bunce, Garrett Maxon
 */

#include "msp.h"
#include "set_DCO.h"
#include "delays.h"
#include "LCD.h"
#include "keypad.h"
#include "combo.h"

#define LOCKED      1
#define UNLOCKED    0

//Mode indicator values
#define NORMAL      1       //NORMAL = Non-Private LOCKED
#define PRIVATE     3       //Privacy enter/exit
#define PRIVLOCK    0xFE    //~LOCKED
#define CLEAR       7       //Entry Clear

void delay_ms(int, int);
void set_DCO(int);
void LCD_INIT(int);
void write_char_LCD(uint8_t, uint8_t, int);
void write_string_LCD(char word[], uint8_t, int);
void clear_LCD(int);
void line_clear_LCD(int, int);
void KEYPAD_INIT();
uint8_t chk_Keypad();

uint8_t chk_Key();
int chk_Password(uint8_t, int);
void LCD_Locked(int);
int LCD_Unlocked(int);
int chk_Privacy(uint8_t, uint8_t);

void main(void)
{
```

-1
Include ALL .c and .h files and organize by filename: LCD.c, LCD.h, Main.c, Delay.c, etc.

```
    WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD; //Stop watchdog timer

    int CLK = 480;                              //48 MHz
    uint8_t lock = LOCKED;                      //Locks the display
    set_DCO(CLK);                               //Sets clock to 48MHz
    LCD_INIT(CLK);                              //Initializes LCD
    KEYPAD_INIT();                              //Initializes keypad

    LCD_Locked(CLK);                            //Sets up the lock screen

    while(1)
    {
        while((lock == LOCKED) || (lock == PRIVLOCK))
        {
            line_clear_LCD(BOTTOM, CLK);    //Clears the password entry line
            lock = chk_Password(lock, CLK); //Checks the next 4 key presses
            line_clear_LCD(BOTTOM, CLK);    //Clears the password entry line
        }

        while(lock == UNLOCKED)
        {
            lock = LCD_Unlocked(CLK);       //Scrolls the unlock message and waits for lock
            delay_ms(200, CLK);             //Waits to avoid holding key press
        }

    }
}

//Checks to see if an alternate modes were activated
//This includes privacy enter, privacy exit, and entry clear
int chk_Mode(uint8_t prevKey, uint8_t curKey, uint8_t mode, int CLK)
{
    if (curKey == K_Ast) {
        if (prevKey == K_Pnd) {
            //Checks to see if privacy mode is already enabled
            if (mode == PRIVLOCK) {
                clear_LCD(CLK);
                LCD_Locked(CLK);                        //Leaves privacy mode
                return PRIVATE;
            }
            else {
                clear_LCD(CLK);
                write_string_LCD("PRIVATE MODE", 0, CLK);   //Enters privacy mode
                return PRIVATE;
            }

        }
        line_clear_LCD(BOTTOM, CLK);                    //Clears previous entries
        return CLEAR;
    }
```

```
    return NORMAL;
}

//
int chk_Password(uint8_t prevMode, int CLK)
{
    uint8_t key1;
    uint8_t key2;
    uint8_t key3;
    uint8_t key4;
    uint8_t addr = 0x40;
    uint8_t newMode;

    key1 = chk_Key();   //Waits for first key entry
    newMode = chk_Mode(K_NP, key1, prevMode, CLK);  //Checks for alternate mode entry
    if (newMode == PRIVATE) {
        delay_ms(200, CLK);
        return ~prevMode;   //Enters or exits privacy mode
    }
    else if (newMode == CLEAR) {
        return prevMode;    //Resets password entry
    }

    //Prints characters entered to LCD if in non-privacy mode
    if (prevMode == NORMAL) {
        write_char_LCD(key1, addr, CLK);
        addr++;
    }
    delay_ms(300, CLK);

    //Repeats process for second key press
    key2 = chk_Key();
    newMode = chk_Mode(key1, key2, CLK);
    if (newMode == PRIVATE) {
        delay_ms(200, CLK);
        return ~prevMode;
    }
    else if (newMode == CLEAR) {
        return prevMode;
    }

    if (prevMode == NORMAL) {
        write_char_LCD(key2, addr, CLK);
        addr++;
    }
    delay_ms(300, CLK);

    //Repeats process for third key press
    key3 = chk_Key();
    newMode = chk_Mode(key2, key3, CLK);
```

```
    if (newMode == PRIVATE) {
        delay_ms(200, CLK);
        return ~prevMode;
    }
    else if (newMode == CLEAR) {
        return prevMode;
    }

    if (prevMode == NORMAL) {
        write_char_LCD(key3, addr, CLK);
        addr++;
    }
    delay_ms(300, CLK);

    //Repeats process for fourth key press
    key4 = chk_Key();
    newMode = chk_Mode(key3, key4, CLK);
    if (newMode == PRIVATE) {
        delay_ms(200, CLK);
        return ~prevMode;
    }
    else if (newMode == CLEAR) {
        return prevMode;
    }

    if (prevMode == NORMAL) {
        write_char_LCD(key4, addr, CLK);
        addr++;
    }
    delay_ms(300, CLK);

    //Compares passcode entered against correct combo stored in outside header file
    if ((key1 == combo1) && (key2 == combo2) && (key3 == combo3) && (key4 == combo4)) {
        return UNLOCKED;
    }
    else {
        line_clear_LCD(BOTTOM, CLK);
        write_string_LCD("INCORRECT CODE", 0x40, CLK);
        delay_ms(500, CLK);
        return prevMode;
    }

}

//Checks for a key press and returns the ASCII character pressed
uint8_t chk_Key()
{
    uint8_t key = 0x10;
    while(key == 0x10)
    {
```

```
        key = chk_Keypad(); //Waits until key is pressed
    }
    return key;
}

//Sets up non-privacy mode lock screen message
void LCD_Locked(int CLK)
{
    uint8_t addr;
    uint8_t letter;
    char word[] = "LOCKED ENTER KEY";

    //Writes lock phrase
    addr = 0x00;
    clear_LCD(CLK);
    write_string_LCD(word, addr, CLK);

    //Sets cursor to bottom row
    letter = 0x10;
    addr   = 0x3F;
    write_char_LCD(letter, addr, CLK);
}

//Scrolls unlock message and waits for lock input
int LCD_Unlocked(int CLK)
{
    int i;
    uint8_t key;
    char word[] = "HELLO WORLD";
    for(i = 0x4F; i >= 0x00; i--) {
        clear_LCD(CLK);
        write_string_LCD(word, i, CLK);
        delay_ms(200, CLK);
        key = chk_Keypad();

        //Locks LCD if "#" is pressed
        if (key == K_Pnd) {
            LCD_Locked(CLK);
            return LOCKED;
        }
        //Handles edge cases
        if(i == 0x00) {
            i = 0x4F;
        }
        else if ((i > 0x0F) && (i < 0x40)) {
            i = 0x0F;
        }
    }
    return UNLOCKED;
}
```

Passcode Header File

```
/*
 * combo.h
 * Holds the passcode for the LCD lock
 *
 * Date: April 14, 2018
 * Author: Zach Bunce, Garrett Maxon
 */

#ifndef COMBO_H_
#define COMBO_H_

//Lock Combo: 9137
#define combo1 0x39
#define combo2 0x31
#define combo3 0x33
#define combo4 0x37

#endif /* COMBO_H_ */
```

-.5
Missing references