Section 1: Description

The application an accomplish every specification in the Lab 2 design document and specification document. First, I aimed to create HAL functions for the LCD screen. I made functions for writing characters to the display, changing the foreground/ background colors, clearing the display, and generating the status messages. The next set of functions I focused on were the buttons to display the status message and clear the display along with button 2 to toggle the baud rate. Then, I programmed the LED as part of the bonus. Finally, I spent most of the time programming the finite state machine for parsing the command and thinking of all possible cases and coding the function such that all cases would be accounted for.

When testing cases in addition to the grading rubric on Canvas, I saw that all of the cases were successful. Button 1 was not debounced, and the display cleared and the status message was displayed in the correct format indicated in the specification. Button 2 was successfully debounced, and the baud rate cycled with a firm press of button 2. I also tested sample commands to be parsed from the design document and the proper characters were displayed on the UART and LCD. I also did take out the lab 2 starter code functionality.

Section 2: Finite State Machine Design

I designed the finite state machine by having 4 states: idle, command, commandF, and commandB. The idle state checks if the character is a plain character or if it is a potential command to change the background or foreground color. If the character is a '#', the state would transition to command. Otherwise, the character would print on both the LCD and UART in the next available cursor position. At command, there would be three possible states to transition to. If the new character being passed into the function were a 'b' or an 'f', there would be a state transition to commandB or commandF respectively. Otherwise, the character '#' would be printed first along with the current character being passed in, and the state transitioning back to idle, waiting for the next character or command. In commandB, the next character at the next function call would be checked to see if it is valid (between 0 and 7, inclusive). If valid, the correct background color would be set by calling the respective function for setting background color. If not valid, the '#' and previous character would be printed, along with the current character. The state would transition to idle for both cases. Similar logic is applied for the commandF state. Below is a diagram that illustrates the finite state machine.

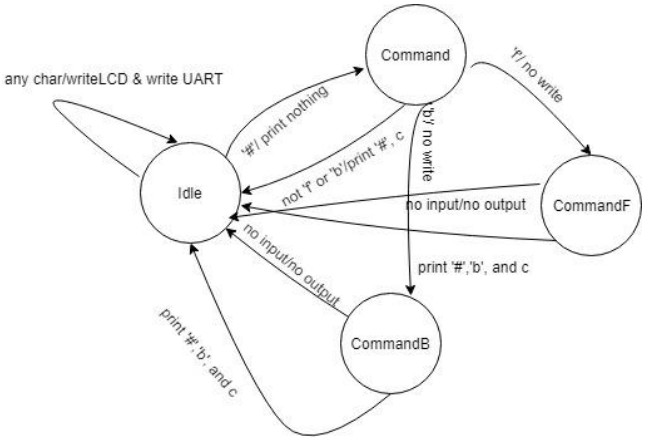

Figure 1. Finite State Machine Design Diagram

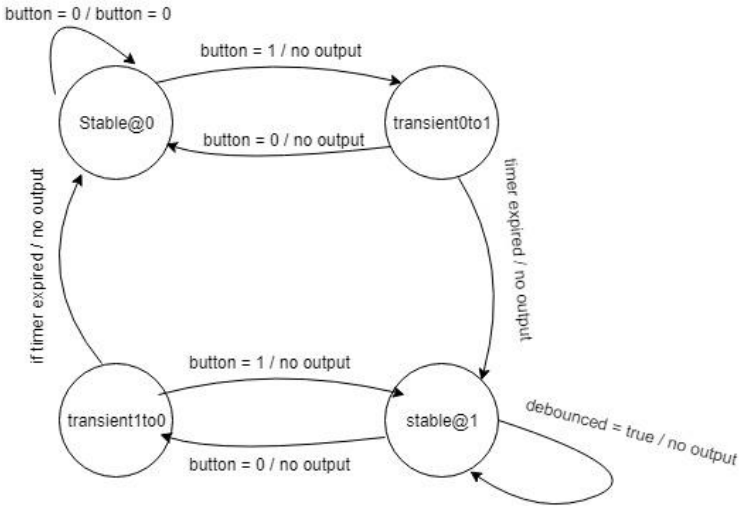

Figure 2. Finite State Machine Design for Button Debouncing

The above figure is the FSM design for the button debouncing HAL function which I used from homework example 4. It is a simple design with four states for where I check whether the button has been pressed and check the timer expiration for the debounce.

Section 3: HAL Design

The graphics portion of the HAL functions were taken from the homework examples. I implemented the following functions for the Graphics. Some were provided within the starter code and some were taken from the respective homework 3 example code, which I was allowed to use:

Graphics:

```
void InitGraphics() – Initalizes the graphics and LCD libraries
```

```
void LCDClearDisplay() – Clears the LCD screen to the most current background color
```

void **LCDDrawChar**(**unsigned** row, **unsigned** col, int8_t c) – writes a character to the proper cursor position being passed in the function to the LCD

void **LCDSetFgColor**() – sets the foreground color using the global variable for the color

void **LCDSetBgColor**() - sets the background color using the global variable for the color

void **write2LCD**(uint8_t inChar) – writes a character to the LCD and maintains proper cursor positions

void **printMessageLCD**() – prints the status message on the LCD

UART

void **InitUART**() – Intializes the UART to be used

bool **UARTHasChar**() – if the UART receives a character, function returns true

uint8_t **UARTGetChar**() – returns the current character received from UART

bool **UARTCanSend**() – the UART register is empty, function returns true

void **UARTPutChar**(uint8_t t) – if the UART is able to send, the character is displayed in the terminal

void **UARTSetBaud**() – changes the baud rate using the global variable for baud rate

void **printMessageUART**() – prints the status message to UART (implemented)

Buttons

bool **BounceFSM**(bool *button) -

void **InitButtonS1**() – initializes button 1

bool **ButtonS1Pressed**() – returns true if button 1 is pressed

void **InitButtonS2**() – initializes button 2

int **ButtonS2Pressed**() – returns true if button is pressed

void **checkButton2Status**(bool *button, bool *prev_button, bool *prev_buttonDebounce, bool *buttonDebounce) – using pass by reference to check whether the button has been pressed and checking for debouncing to correctly toggle baud rates

LEDs

void **InitRedLED**() – initializes the red LED

void **RedLEDToggle**() – toggles the red LED

void **ColorLEDSet**(color_t t) – sets the color LED to whatever enum t holds

InitColorLED() – initializes the color LEDS on the booster

void LEDchange(uint8_t character) – bonus function to toggle proper LED depending on character

Timer

void InitTimerDebounce() – initialize debounce timer

void TimerDebounceStartOneShot() – starts debounce timer

int TimerDebounceExpiredOneShot() – checks if debounce timer has expired

void Init200msTimer() – initialize 200 ms timer

void Timer200msStartOneShot() – starts a timer for 200 ms

int Timer200msExpiredOneShot() – returns 1 if timer has expired

Other Functions

void parseCommand(uint8_t c) – takes in a character from UART and goes through the FSM

I wrote the following functions:

void parseCommand(uint8_t c) – takes in a character from UART and goes through the FSM

void LEDchange(uint8_t character) – bonus function to toggle proper LED depending on character

void checkButton2Status(bool *button, bool *prev_button, bool *prev_buttonDebounce, bool *buttonDebounce) – using pass by reference to check whether the button has been pressed and checking for debouncing to correctly toggle baud rates

void printMessageUART() – prints the status message to UART (implemented)

void write2LCD(uint8_t inChar) – writes a character to the LCD and maintains proper cursor positions

void printMessageLCD() – prints the status message on the LCD

In addition to these functions, I used all of my homework 3 functions to help with this lab. They are listed above with the HAL functions.