Ragon005 – buss0222 – Lab 4 report – 7 seg Display and Keypad.

This was a very informative lab. The idea behind the keypad column and row checking was very insightful. I feel like I got a lot out of this lab with having to create the scanning function and bit switching. There were quite a lot of lab components to build. When I first built the circuit, I had issues with hooking up the output pins to the wrong parts of the seven-segment display. I had to debug it using the wire and no chip method via the lab manual. After this was done I began to create the code for displaying the keypad letters on the clock. This part of the lab went smooth. I mapped out each segment of the display to the RB pin that hosted it. I then used an enum table filled with the bits required to turn those segments high. This was easily passed around to create any display I wanted. I could simply call left or right for an anode bit, along with the segment. I mapped out the segment display to OR each enum parameter in the display function. Moving passed this, Jennifer and I began to create the code for the read keypad function. This part of the lab took some time to get right. While at first we were able to create the inputs of the keypad to one side or the other we were unable to get the shifting correct. Our initial design had no check that would check to stop our read keypad function from placing multiple items of the same press into the array that stored our look ups for the right and left displays. This was then debugged after I consulted the professor on what exactly the problem could be. After we that we initially set fourth to create the function like the lastKey feature in the discussion code. Using the same logic as the professor we were able to get the display to shift properly. Our read keypad function was initially setup to use the algorithm row*4+col. Even though this did work for us on some level I had initially setup the LATB 15:12 pins wrong and it was causing an issue of a button pressed in a row to display any element in that column. I tore the read keypad code apart and started from scratch. I began the new read key pad function to set each column as an individual for case by case basis using LATBbits. This was as followed:

0111, 1011, 1101, and 1110. Through this and the logic analyzer I was able to see my mistake of not initially setting RB15:12 to 1111. This basically solved the issue. Instead of switching back to the row*4+col, I left the case by case check in the read keypad function. Prior to fixing the code there were nested if statements for each case by case of the read key that checked if the button was pressed or not. I had forgot to remove all of these for our lab demo. Therefore, we had an issue of the ‘d’ button being pressed and then display not being updated. This was easily fixed and our demo went smooth.

Relevant Code:

```
    LATBbits.LATB12 = 0;

    LATBbits.LATB13 = 1;

    LATBbits.LATB14 = 1;

    LATBbits.LATB15 = 1;


static unsigned int lastKey[] = {17,17,17,17};



if(keyToCheck != lastKey[row])

    {
        lastKey[row] = keyToCheck;

        if(keyToCheck != 17)

        {

            set(keyToCheck);

        }

    }
```

And:

```
LATB = 0xf000;
```