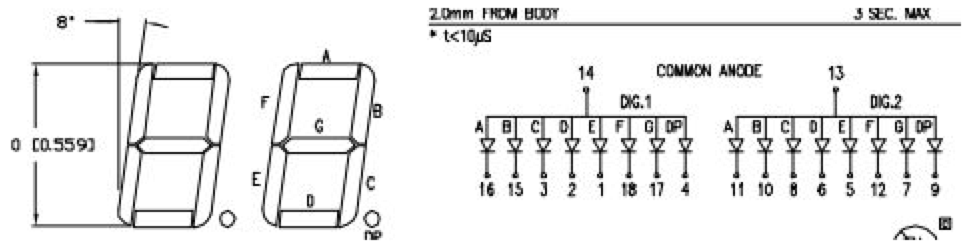
Lab report 3

The purpose of this lab is to demonstrate the working concepts of seven segment display using pic 24 MCU and a 4-4 matrix keypad as tools as well as troubleshooting. The first part done for this lab was the seven segment display. In order to display the Arabic numbers, each segment that needs to light up has to be turned on. The display is made up of LEDs which conform to the principle of forward biasing, meaning that there needs to be high signal emitted to the positive side and low signal to the negative. By using bitwise operation controlling the signal pulse from ports, every number was able to be displayed correctly. An example is to display ‘1’ on the right display, in order to show the correct output, the “B” and “C” segments which are No. 10 and 8 LEDs inside the right display need to be operating. The code to achieve that is shown here with the standard of the circuit schematic.



```
LATA = 0xf000;
if (myChar=='1'){
        LATB|=0x0a7f;//setting RB7 and RB8 0 and RB10 0, and the rest except for the
most significant 4 bits 1. 0b0000101001111111
}
```

The next part of the lab was the implementation of the connection between the keypad and the display. The algorithm used for this purpose was cycling through each of the four rows with detection of the user's pressing from the perspective of the column. The following code demonstrates this purpose.

```
char readKeyPadRAW(void){
   char keyArray[]={'1','2','3','A','4','5','6','b','7','8','9','C','E','0','F','d'};
   int rowAct[4]={0xe000,0xd000,0xb000,0x7000};
   int row;
   int col,keyIndex;
   for(row=0;row<4;row++){
      LATB&=0x0fff;
      LATB|=rowAct[row];
      xie_wait_50us();//debounce
```

```
        col=column();
        if(col!=-1){
            break;
        }
    }
    if(col==-1){
        return 'n';
    }else{
        keyIndex=row*4+col;
        return keyArray[keyIndex];
    }
}
```

Here, nokey variable was set the char ‘n’ which doesn’t exist on the display(key array). Whenever there is no detection of pressing on any column, it means that the keypad is currently in idle state and the nokey variable will be returned. Whenever it appears detection in column, the step is to capture the row number of that detection and with a simple pattern calculation to return the correct key from the array that is the crossing of the detected row and column. This method was tried and tested correctly shown on the display. However, when continuing the next step of the lab which is to “For every key press, shift the rightmost displayed value to the left seven-segment digit (most significant), then display the key pressed on the rightmost digit(least significant),” it appeared that the two displays would both get the signal of a new key being pressed, which then would show the same demonstration. To avoid that from happening, the debounce waiting was added in order to stabilize the pressing of the key for 50us. This achieved the goal of displaying different numbers on two displays but the most significant display would blind continuously with high frequency. Delay function was first tried after the display of that display but failed with no significant difference. Another method was then implemented for the troubleshooting of this occurrence, another variable “pressedKey” was created inside the main loop to remember the “new” key being pressed. The code is shown here,

```
        pressedKey = Key;
        Key =readKeyPadRAW();
        if ((Key !='n') && (new!=pressedKey)){
            old=new;
            new=Key;
        }
        showChar7seg(old,1);
```

```
    xie_wait_1ms();
    xie_wait_1ms();
    xie_wait_1ms();
    xie_wait_1ms();
    xie_wait_1ms();
    showChar7seg(new,0);
    xie_wait_1ms();
    xie_wait_1ms();
    xie_wait_1ms();
    xie_wait_1ms();
    xie_wait_1ms();
```

Whenever the detected key in the crossing of the row and column is not nokey AND the new key being pressed is not equal to the last key being pressed, updating and shifting of display can only be granted. This solved the issue of blinking since the condition became more strict for display updating which avoided the most significant key from keeping updating itself to the same of itself, which caused it to keep blinking. The whole experiment was achieved successfully with all implementations described, and a demo was presented.