Memo to: Mr. Cory Mettler
From: Julian Rechsteiner, Bennett Christenson
Date: April 06, 2021
Regarding: EELE 465, Lab 5 – Analog to Digital Temperature Readings

Summary:
The purpose of this lab was to use an MSP430-FR2355 microprocessor to read data from a LM19 temperature sensor, take a custom moving average of data from this sensor based off user input, and display the data on the LCD via I2C master-slave communication.

Setup:
The keypad was connected to the master device using ports 1.0-1.7, while the I2C clock and data lines utilized ports 4.7 and 4.6 respectively, Figure 1. The LM19 temperature sensor's Vout was connected to the Master via port 5.0 and was powered with the MSP 3.3V supply and grounded using the MSP ground. Two pull-up resistors were added to these lines to achieve a consistent logic high. These lines were then connected to ports 1.3 and 1.2 respectively to each slave device. Slave device 1 interfaced with the LED bar through ports 1.0, 1.1, and 1.4-1.7, along with ports 2.7 and 2.6. A resistor of 101-ohm sat in the middle to reduce the current flow. Slave device 2 interfaced with the LCD display through ports 1.7-1.4 as data ports, 2.6 as the RS port, and 2.7 as the enable port. The LCD was powered by a 3.3-volt supply through a potentiometer output to the display power and the logic was powered using just the 3.3-volt supply.

Step 1 – LCD Reset Condition:
The LCD Reset condition was implemented successfully using an if statement to catch if the ascii code for asterisk was sent. When this condition was caught, the LCD slave would then assert a function that drove the LCD to its original state, cleared of any temperature inputs from the master.

Step 2 – Temperature Data Sampling:
To accomplish a goal of acquiring three samples of temperature values every second, a timer interrupt was created. It was configured to use the A-Clock which has a frequency of 32.768 KHz, which was divided by 8 and set to have a bit length of 12. These caused the timer to overflow at 1 second. The timer was then set to up mode and set to overflow at a third of the normal overflow value, thus creating a .33 second overflow, Figure 4. In the corresponding interrupt service routine to this timer, the data from the LM19 would be read and stored in an array. When the array became fully populated with data based on the user input, the index would then go back to the beginning and override the oldest value with a new value.

Step 3 – Moving Average:
The moving average was programmed to be calculated only once the array with the LM19 data was populated with the correct number of data points, Figure 2. For example, if six data points were selected via the keypad, the moving average would only be calculated once there were six ADC values in the array.

Step 4 – Conversions:
Once the average of the ADC values was calculated, many conversions had to take place to output the correct data to the LCD, Figure 3. Initially, the ADC average was converted to degrees in Celsius, Equation 1. The temperature was then converted to Kelvin by adding 273 to the value in Celsius. Once the temperature values were determined, they were converted to ASCII characters and moved to an array which was sent to the LCD slave module using I2C. Modulus division was performed on each temperature value to obtain each digit of the number, which subsequently was converted to ASCII.

$$TempC_{avg} = (-0.284 * avgADCvalue) + 289.89; \quad [1]$$

Conclusion:
The focus of this project was to read values from the LM19 temperature sensor and output a moving average, determined by user input, to the LCD. The Master would take in user input and set the moving average number to a user defined number. Then, every 333ms, the master would take in values from the LM19 and store the number into an array. Once the array was sufficiently populated, the master would take the average of the values. The master would

then convert those numbers into Celsius and Fahrenheit, and then those into ASCII representations of those numbers. Those ASCII characters would then be sent to the LCD slave to be displayed.

Lessons Learned:

- Turn Optimizations off if delays aren't working properly
- LM-19 is polarized – connection orientation matters
- Don't get too hung up on the operation of any particular device for too long before moving on
- Sometimes the easiest path is often the best path

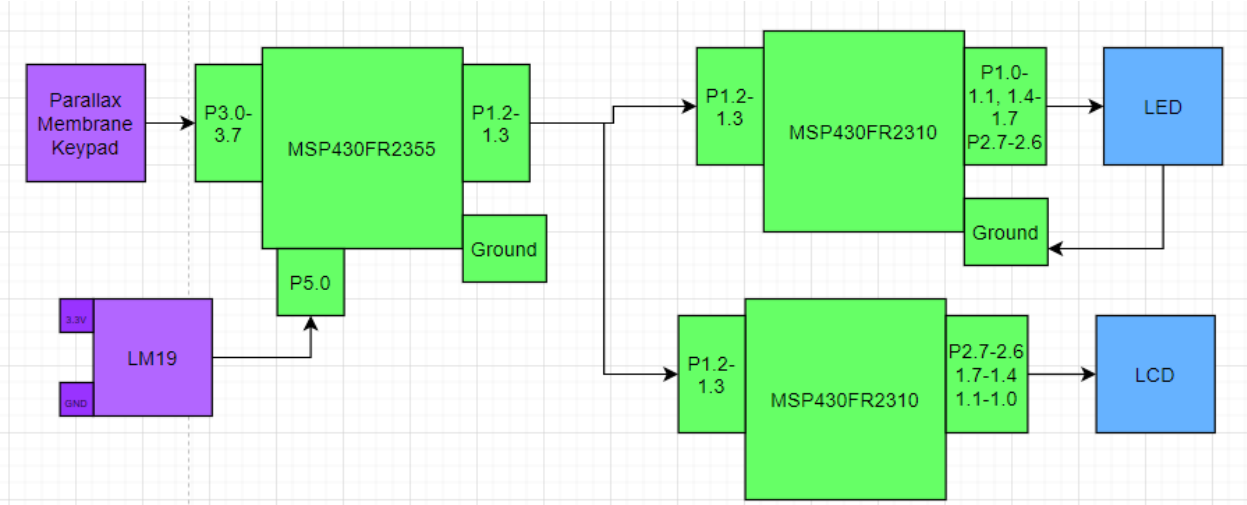

Figure 1: Circuit Diagram

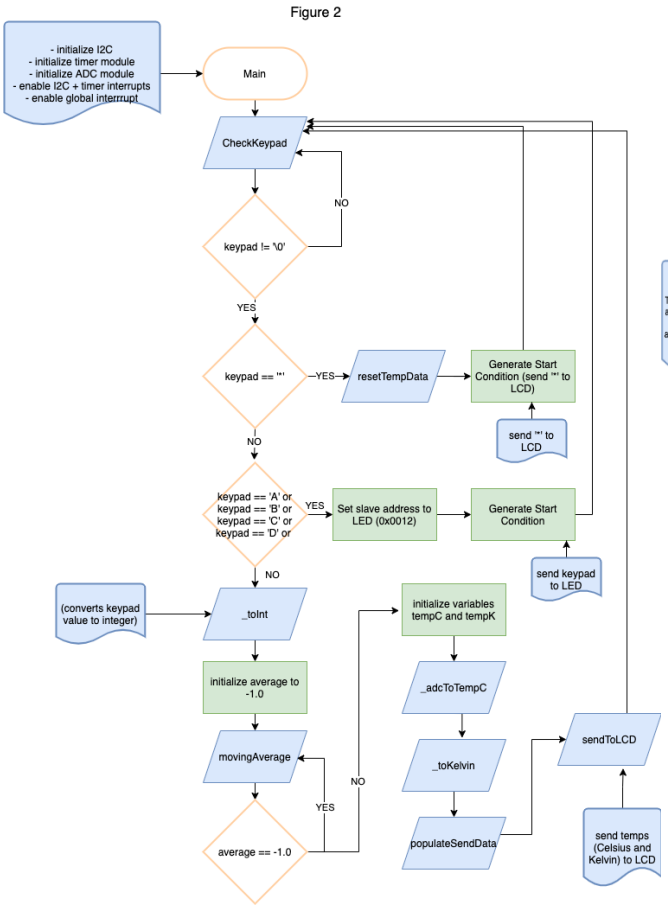

Figure 2: Master Flowchart

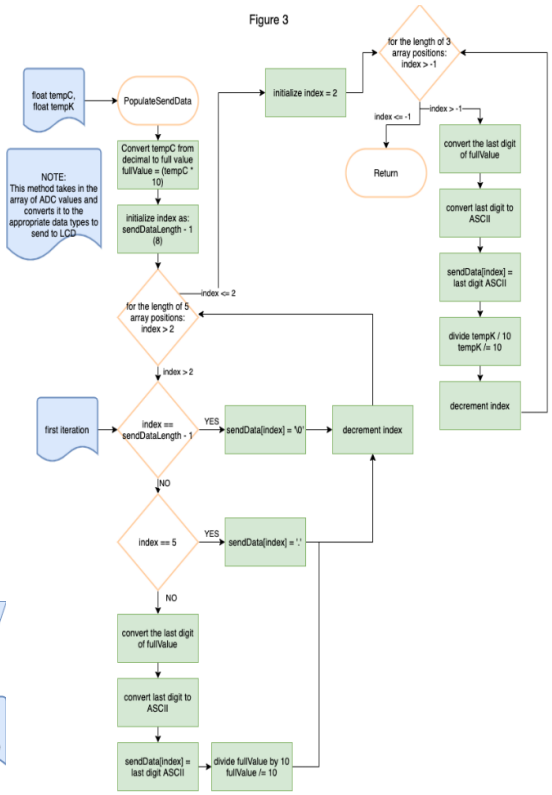

Figure 3: Data Population Flowchart

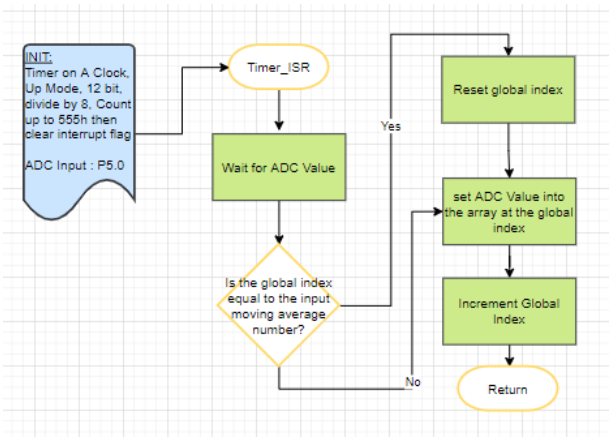

Figure 4: Timer ISR